UNIVERSIDADE FEDERAL DE SANTA CATARINA - CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

Desenvolvimento de um Modelo de Avaliação da Estética Visual de Interfaces de Usuários de Aplicativos Usando *Deep Learning*

Osvaldo Paulo Heiderscheidt Roberge Martins

Florianópolis
2019

Universidade Federal de Santa Catarina
Departamento de Informática e Estatística

Desenvolvimento de um Modelo para Avaliação da Estética Visual de Interfaces de Usuários de Aplicativos Usando *Deep Learning*

Trabalho de Conclusão de Curso de Graduação em Ciências da Computação, do Departamento de Informática e Estatística, do Centro Tecnológico da Universidade Federal de Santa Catarina, requisito parcial à obtenção do título de Bacharel em Ciências da Computação.

Autor: Osvaldo Paulo Heiderscheidt Roberge Martins

Orientadora: Prof.ª Dr.ª rer. nat. Christiane Gresse von Wangenheim, PMP

Florianópolis
2019

Osvaldo Paulo Heiderscheidt Roberge Martins

Desenvolvimento de um Modelo para Avaliação da Estética Visual de Interfaces de Usuários de Aplicativos Usando *Deep Learning*

Trabalho de conclusão de curso submetido ao Departamento de Informática e Estatística da Universidade Federal de Santa Catarina para a obtenção do Grau de Bacharelado em Ciências da Computação.

Florianópolis, 9 de dezembro de 2019

Prof.ª Dr.ª rer. nat. Christiane Gresse von Wangenheim, PMP
Orientadora
Universidade Federal de Santa Catarina

Banca Examinadora:

Prof. Dr. Jean Carlo R. Hauck
Avaliador
Universidade Federal de Santa Catarina

Prof. Dr. Mauro Roisenberg
Avaliador
Universidade Federal de Santa Catarina

RESUMO

O desenvolvimento tecnológico dos últimos anos tornou a computação indispensável no cotidiano da população, tanto na carreira profissional quanto na vida pessoal. Devido a sua grande importância foram criadas várias iniciativas de ensino de computação já na educação básica. O ensino de computação para esses jovens, utilizando ambientes de desenvolvimento como o App Inventor, auxilia os alunos a aprender competências básicas de programação e pensamento computacional. Além da programação em si, outro aspecto importante é o design das interfaces de usuário de apps visando a sua usabilidade. Por isso, como parte do processo de ensino-aprendizagem torna-se importante, também, o ensino e a avaliação da estética visual dos aplicativos criados pelos alunos. Atualmente esta avaliação se dá de maneira manual, exigindo tempo, esforço e preparação por parte dos instrutores, pontos estes que até mesmo dificultam a adoção do ensino de computação em escolas brasileiras. Nesse contexto, o objetivo deste trabalho é o desenvolvimento de um modelo utilizando *deep learning* para avaliar automaticamente a estética visual de projetos desenvolvidos com o ambiente App Inventor, a ser utilizada em unidades instrucionais para ensinar a computação na educação básica. Como resultado é desenvolvido um modelo de regressão, “Appsthetics”, para avaliar a estética visual de interfaces de usuários a partir de uma *screenshot* da mesma. A análise de desempenho demonstra resultado aceitável (erro quadrático médio = 0.051369) e a demonstração da correlação/concordância com os avaliadores humanos por meio da correlação Pearson (r = 0.74) e a análise Bland & Altman (0.0051) também indicam uma relação aceitável entre as predições e os valores verdades. O modelo desenvolvido é posteriormente integrado à ferramenta CodeMaster, para incluir então uma forma de avaliar o aprendizado de alunos em relação a estética visual de interfaces em seus *apps*.

Com isto espera-se facilitar a avaliação da aprendizagem dos alunos, contribuindo dessa maneira ao ensino de computação mais amplamente nas escolas brasileiras.

Palavras chave: Estética Visual, App Inventor, Design de Interface de Usuário, Educação Básica, *Deep Learning*.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1. INTRODUÇÃO

1.1 Contextualização

Ao longo das últimas décadas, computadores transformaram profundamente, de várias formas, o mundo e o modo de trabalho. Como resultado, computação e as tecnologias que ela habilita estão no centro da nossa economia e no modo como vivemos nossas vidas (CSTA, 2016). Para ser um cidadão bem instruído em um mundo intensamente informatizado e estar preparado para carreiras no século XXI, os estudantes têm que ter um entendimento claro dos princípios e práticas de computação (CSTA, 2016).

O ensino de computação na Educação Básica é uma iniciativa para popularizar a competência de computação, proporcionando dessa forma um conhecimento básico de computação para toda a população como uma das habilidades importantes no século XXI. O ensino desses princípios e práticas tem benefícios que vão além de uma formação profissional. Estudantes de computação aprendem raciocínio lógico, pensamento algorítmico, concepção e resolução de problemas estruturados (CSTA, 2016). Estudar computação pode preparar um estudante para uma carreira em muitas áreas, tanto dentro quanto fora da computação (CSTA, 2016).

Uma alternativa para ensinar computação na educação básica é o uso de ambientes de programação visual baseados em blocos como o App Inventor, que conta com com mais de 400.000 usuários ativos mensalmente, espalhados entre 195 países (MIT, 2017). Com o App Inventor é possível criar jogos, animações e até mesmo aplicativos mais avançados, usando execução condicional, eventos, operadores matemáticos e *booleanos*, listas entre outros recursos. O ambiente visual do App Inventor é amplamente convidativo tanto para crianças como para adultos, além de ser uma plataforma para sério estudo de computação que busca democratizar o desenvolvimento de software capacitando todas as pessoas, especialmente os jovens, para passarem de consumidores a criadores de novas tecnologias (MIT, 2017).

No contexto de desenvolvimento de aplicativos, para se desenvolver apps bem-sucedidos, um dos critérios de sucesso é a sua estética visual. Usuários gostam de interagir com interfaces com design esteticamente agradáveis, pois elas satisfazem seus sentidos (NIKOLOV, 2017). A estética visual é definida por elementos-chave como cor, forma, padrão, textura, peso visual, equilíbrio, escala, linha, proximidade e movimento,

que necessitam de bom e harmonioso uso para se obter bons níveis de estética visual (NIKOLOV, 2017). Atualmente, usuários de aplicativos móveis vêm se tornando cada vez mais seletivos sobre suas escolhas, descartando imediatamente aqueles que não lhe agradam visualmente (MINIUKOVICH; ANGELI, 2015), o que agrega ainda mais incentivos para o bom ensino de boas práticas de design de interfaces de usuários. Portanto, o ensino sobre estética visual no contexto de design de interfaces já deveria ser parte do conteúdo ensinado na educação básica em cursos de desenvolvimento de aplicativos.

No processo de ensino-aprendizagem, para que seja bem-sucedido, é muito importante o fornecimento de *feedback* ao aluno (HATTIE; TIMPERLEY, 2007), o que pode ser feito por meio de avaliações de desempenho com base nos próprios artefatos criados por ele. Assim, por exemplo, em cursos de desenvolvimento de aplicativos os professores podem avaliar a estética visual dos aplicativos criados pelos alunos (FERREIRA, 2019). Porém, apesar da avaliação ser, por natureza, um processo objetivo, o julgamento humano, ainda mais em se tratando de estética, se torna alvo de influências e preferências variáveis entre cada pessoa, o que torna este um processo subjetivo onde não há uma "verdade absoluta" (ZEN; VANDERDONCKT, 2016), (GRESSE VON WANGENHEIM et al., 2018b). Assim, uma alternativa pode ser automatizar o processo de avaliação da aprendizagem. Nesse contexto, técnicas de *deep learning* são utilizadas com sucesso para a avaliação automática da estética de imagens (LU et al., 2015). Essas técnicas fazem uso de modelos de redes neurais convolucionais profundas (DCNN[1]) e vêm apresentando resultados muito similares ao que se esperaria de uma avaliação humana (MALU et al., 2017). Contudo, atualmente as pesquisas estão focadas na avaliação de fotografia e *websites*, ainda não existindo ferramentas voltadas à avaliação estética de interfaces de usuário de aplicativos.

[1] do inglês *Deep Convolutional Neural Networks*

1.2 Objetivos

Objetivo geral

O objetivo geral é desenvolver um modelo para a avaliação da estética visual de interfaces de usuário de aplicativos usando técnicas de *deep learning* para criar, treinar e testar uma rede neural para que seja capaz de realizar tal tarefa, visando facilitar a avaliação da aprendizagem dos alunos referente à estética visual de apps Android. O modelo desenvolvido é integrado à ferramenta CodeMaster (GRESSE VON WANGENHEIM et al., 2018a), para incluir uma forma de avaliar o aprendizado de alunos em relação a estética visual de interfaces em seus *apps*.

Objetivos Específicos

O1. Analisar a fundamentação teórica sobre estética visual de interfaces de usuário e *deep learning*.

O2. Analisar o estado da arte em relação a análise automática da estética visual de interfaces de apps;

O3. Desenvolver e testar um modelo/arquitetura de rede neural utilizando *deep learning*;

O4. Integrar a funcionalidade da avaliação automatizada ao sistema CodeMaster.

Premissas e restrições

O trabalho é realizado de acordo com o regulamento vigente do Departamento de Informática e Estatística (INE – UFSC) em relação aos Trabalhos de Conclusão de Curso. O modelo proposto tem como foco a análise e avaliação da estética visual de interfaces de usuário, não abordando outras qualidades. O trabalho também foca na análise de aplicativos Android não abordando interfaces de apps de outras plataformas. A automatização enfoca em projetos desenvolvidos com a ferramenta de desenvolvimento App Inventor no ensino de computação na educação básica (especificamente ensino fundamental II).

1.3 Metodologia de Pesquisa

Nessa pesquisa é usada uma abordagem multi-método. A metodologia de pesquisa utilizada neste trabalho é dividida nas seguintes etapas.

Etapa 1 – Fundamentação teórica

Atividade focada em estudar, analisar e sintetizar os conceitos principais e a teoria referente aos temas a serem abordados neste trabalho. Nesta etapa é apresentada a fundamentação teórica utilizando a metodologia de revisão narrativa (CORDEIRO et al., 2007), e são realizadas as seguintes atividades:

A1.1 – Análise teórica sobre estática visual de interfaces de usuário;

A1.2 – Análise teórica sobre *deep learning*.

Etapa 2 – Estado da arte

Nesta etapa é realizado um mapeamento sistemático da literatura seguindo o processo proposto por Petersen et al. (2008) para identificar e analisar modelos de análise automatizado da estética visual de interfaces de usuário de apps atualmente sendo utilizados. Esta etapa é dividida nas seguintes atividades:

A2.1 – Definição do protocolo de busca;

A2.2 – Execução da busca e seleção de artigos relevantes;

A2.3 – Extração e análise de informações relevantes.

Etapa 3 – Desenvolvimento

Nesta etapa é desenvolvido um modelo para análise da estética visual de interfaces de apps criados no App Inventor, seguindo um processo de desenvolvimento de redes neurais/*deep learning* (KIERSKI, 2017; SHUAI, 2017; POLYZOTIS et al., 2017). Esta etapa é dividida nas seguintes atividades, onde, com exceção de A3.1, foram realizadas por meio de diversas iterações:

A3.1 – Análise de requisitos;

A3.2 – Coleta de dados;

A3.3 – Preparação de conjunto de dados;

A3.4 – Seleção de um modelo de rede neural;

A3.5 – Treinamento da rede neural;

A3.6 – Avaliação da rede neural;

A3.7 – Afinação de parâmetros;

A3.8 – Predição/Inferência.

Etapa 4 – Integração

Nesta etapa é integrada essa funcionalidade no sistema CodeMaster (GRESSE VON WANGENHEIM et al., 2018a) seguindo um processo de engenharia de software proposto por Pressman (2016). Esta etapa é dividida nas seguintes atividades:

A4.1 – Análise de requisitos;

A4.2 – Modelagem da arquitetura do sistema;

A4.3 – Modelagem detalhada e implementação;

A4.4 – Testes do sistema.

1.4 Estrutura do Documento

No capítulo 2 é apresentada a fundamentação teórica dos conceitos necessários que sustentam a proposta deste trabalho. No capítulo 3 é apresentada a situação atual em que se encontram os trabalhos e propostas existentes na área de avaliação automatizada da estética da interface de usuário em aplicativos móveis utilizando técnicas de *Deep Learning*. O capítulo 4 apresenta os resultados criados no desenvolvimento do modelo “Appsthetics”. O capítulo 5 apresenta os resultados e comparações das análises de desempenho do modelo desenvolvido. O capítulo 6 apresenta o trabalho de integração do modelo ao sistema CodeMaster. Para finalizar, o capítulo 7 apresenta a conclusão do presente trabalho.

2. FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são apresentados conceitos essenciais, relevantes ao trabalho sendo desenvolvido. Iniciando por uma análise teórica sobre a estética visual no âmbito de interfaces de usuário, seguido por uma análise teórica sobre inteligência artificial, com foco em *deep learning*.

2.1 Estética

A estética, apesar de tradicionalmente focar na percepção e discriminação da beleza, prazer e emoção, pode ser cientificamente definida como o estudo do sensorial, ou valores sensório-emocionais, algumas vezes chamados de julgamentos de sentimento e gosto (ZANGWILL, 2019). A estética se refere à experiência sensorial que um produto provoca, e até que ponto essa experiência se encaixa com objetivos pessoais (THUERING; MAHLKE, 2007). O efeito da estética visual de uma interface revela-se no comportamento de uso, escolhas entre alternativas, bem como nos julgamentos que os usuários fazem sobre a interface - depois de um primeiro período curto de visualização, bem como após diversas interações (PAJUSALU, 2012).

A criação de artefatos, sejam físicos ou digitais, é, então, fortemente influenciada por algum senso estético, que será responsável pelos sentimentos imediatos invocados pela experiência de um usuário com o artefato através de algum sistema sensorial (ULRICH, 2006). É evidente a importância da estética no design; se todas as outras opções são iguais, a maioria dos usuários irá preferir um artefato belo ao invés de um feio, mesmo em domínios funcionais como, por exemplo, instrumentos científicos (ULRICH, 2006). Portanto, beleza pode até mesmo ser pensada como mais um atributo na avaliação de preferências do usuário, juntamente com durabilidade, facilidade de uso, custo, segurança, etc. A qualidade estética de um artefato é um fator importante para prover uma experiência satisfatória, que é um dos motivos maiores do design (ULRICH, 2006).

2.2 Teoria do Design

A teoria do design envolve os fundamentos e princípios da criação de comunicação visual e de todos os tipos de arte, lidando com a forma como vemos e percebemos a informação

visual, separando as ideias de estilo, gosto e tendência dos princípios universais da estética que são comuns a todas as pessoas (LUNDGREN, 2018). Ela envolve uma compreensão dos elementos tangíveis, incluindo forma, espaço, proporção, cor, escala, textura, estrutura, composição, linha, forma e volume, bem como organizá-los para alcançar equilíbrio, ritmo, padrão, hierarquia, ênfase e unidade (LUNDGREN, 2018).

Em Interação Humano-Computador (HCI[2]), uma interface de usuário se trata de uma parte de um computador e seu software que pessoas podem ver, ouvir, tocar, conversar ou entender, ou dirigir (GALITZ, 2007). Neste contexto, o design tem grande importância na criação de interfaces de usuário: se trata do estudo, planejamento, e design de como pessoas e computadores trabalham em conjunto para que as necessidades do usuário sejam satisfeitas da maneira mais eficiente possível (GALITZ, 2007). Vários fatores são levados em consideração neste processo: o que pessoas querem e esperam, que limitações físicas e habilidades elas possuem, suas percepções, e o que elas acham agradável e atraente (GALITZ, 2007).

Para obter um design de interface com boa usabilidade e estética visual visa-se a abordagem de aspectos de uma linguagem visual de interfaces (LIDWELL, 2009) (SCHLATTER; LEVINSON, 2013), organizando-a em **meta-princípios** e **ferramentas de usabilidade visual**.

Tabela 1: Organização da linguagem visual de uma interface (SCHLATTER; LEVINSON, 2013)

Meta-princípios	**Elementos[3] do Design Visual**
Consistência Hierarquia Personalidade	Layout Cor Tipografia Imagens Controles e *affordances*

[2] do inglês Human-Computer Interaction

[3] Schlatter e Levinson (2013) se referem a estes elementos como “ferramentas” em seu trabalho, dando a justificativa de que designers e desenvolvedores os manipulam de modo a comunicar mensagens e funcionalidades, dando a ideia de um comportamento de uma ferramenta.

2.2.1 Meta-princípios do Design Visual

Estes são os princípios de maior influência, maior importância no design visual de uma aplicação. Mesmo com inovações e desenvolvimento da tecnologia, estes conceitos permanecem presentes em cada etapa, em cada elemento, do design visual de aplicações.

Consistência se trata de definir e manter expectativas por meio do uso de elementos que as pessoas já estão familiarizadas com. Expectativas são criadas pelo o que as pessoas vêm na tela, bem como o que elas já viram no passado (SCHLATTER; LEVINSON, 2013).

A Figura 1 ilustra três exemplos de organizações comuns em telas de *smartphones* que usuários estão acostumados a ver e correspondem a um bom exemplo de consistência.

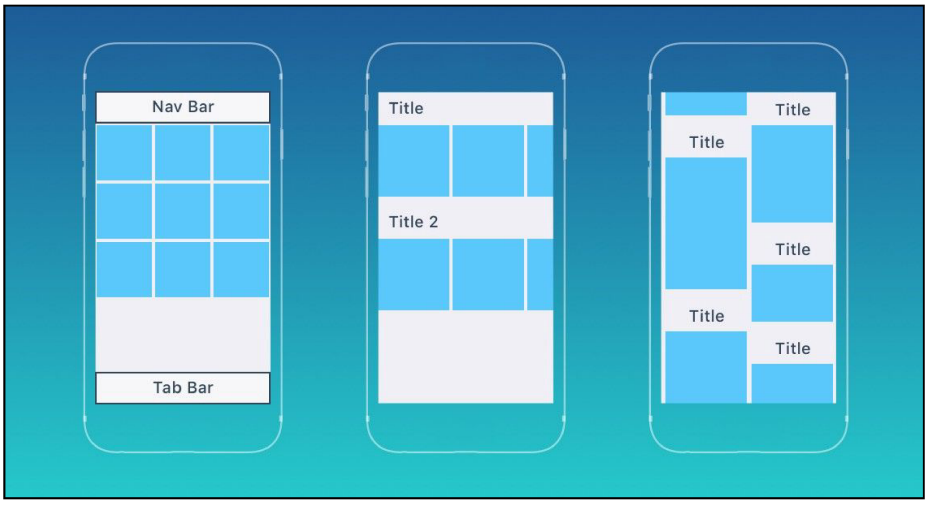

Figura 1: Posicionamento consistente de elementos na tela (adaptado a partir de SMITH, 2018a).

Os seguintes tipos de consistência constituem este princípio (SCHLATTER; LEVINSON, 2013):

- Visual: elementos perceptíveis pelo usuário.
- Funcional: controles da aplicação, como botões e ações do usuário.
- Interna: a combinação do visual e do funcional na sua aplicação.
- Externa: tipo de consistência que é alcançada através de múltiplas aplicações, quando o usuário é capaz de reusar conhecimentos obtidos em uma experiência prévia em uma nova aplicação.

A presença de inconsistências pode não deixar claro ao usuário quais ações estão disponíveis a ele, confundindo-o e prejudicando sua experiência com o aplicativo, como mostra a Figura 2.

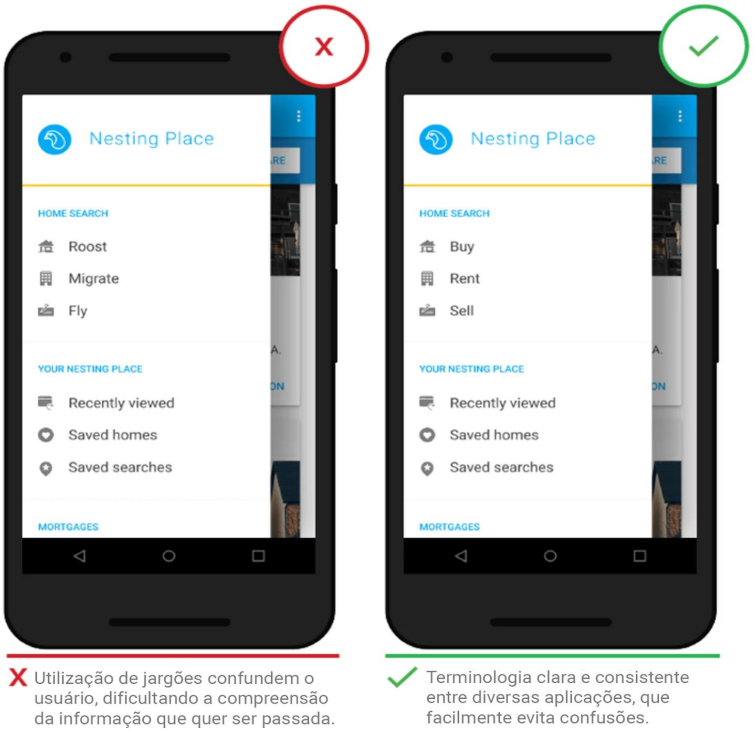

Figura 2: Exemplo de inconsistência no uso de termos para descrever ações básicas em uma aplicação (adaptado a partir de BABICH, 2019).

Hierarquia visual é a percepção e interpretação da importância relativa de objetos. No design de aplicações estes objetos são os elementos presentes em uma tela: posição, tamanho, cor e tipos de controle de interfaces (botões, links, etc.) (SCHLATTER; LEVINSON, 2013). Quando fazem parte da interface como um todo, também afetam como o usuário interpreta a informação que lhe está sendo passada. As características de cada elemento e sua justaposição comunicam informações sobre suas prioridades.

A Figura 3 ilustra um exemplo comparativo entre duas interfaces, uma com bom uso de hierarquia e outra não; podemos ver que há informações demais na tela da esquerda e uma falta de organização por “ordem de grandeza” entre os elementos. Já a tela à direita podemos ver que o logo se tornou a parte principal de cada item na lista e o espaçamento maior entre cada elemento elimina a sensação de “sufoco” por excesso de informação.

Figura 3: Comparativo de hierarquia entre interfaces (adaptado a partir de KONIOR, MARTYNOWSKI, 2016).

Na prática, há falta de hierarquia quando objetos na tela atraem os olhos do usuário igualmente, em outras palavras, quando nada se sobressai na visão do usuário (SCHLATTER; LEVINSON, 2013). Se todos os elementos em uma página ou tela possuírem um mesmo peso visual, haverá uma dificuldade de se fazer sentido nas informações e na navegação da mesma.

As principais características da hierarquia (SCHLATTER; LEVINSON, 2013):

- Contraste: sobre a criação de diferenças visíveis entre elementos.
- Posição: aspectos de posicionamento de objetos na tela que afetam seu contraste e a percepção hierárquica (agrupamento, proximidade, similaridade…).
- Tratamento: o estilo visual aplicado aos elementos afetam sua importância na percepção do usuário (tamanho, cores, detalhes, …).

Aparência, comportamento, e satisfação são fatores importantes em se tratando de como pessoas julgam uma aplicação. Como este trabalho não trata aspectos funcionais de comportamento dos Apps, enfocaremos na **personalidade visual** da aplicação. O que pessoas veem por meio do *layout*, da cor, da tipografia, das imagens e controles afetam não apenas suas primeiras impressões, mas também como usam e consideram a aplicação (SCHLATTER; LEVINSON, 2013).

Norman (2005) fornece um *framework* para a personalidade de produtos:

- Design visceral mapeia para aparência.
- Design *de comportamento* representa o prazer e efetividade de uso.
- Design reflexivo relaciona-se à própria imagem, satisfação pessoal, e memórias.

É importante ressaltar que não se cria “personalidade” (SCHLATTER; LEVINSON, 2013). Pessoas determinam personalidade com base em características interpretadas de padrões detectados através da percepção e experiência ao fazerem uso das aplicações (SCHLATTER; LEVINSON, 2013).

2.2.2 Elementos do Design Visual

Os elementos do design visual são manipulados por desenvolvedores para comunicar uma certa mensagem ao usuário (SCHLATTER; LEVINSON, 2013). Aqui são apresentados os principais elementos e como estes afetam a interpretação de cada usuário.

Layout se trata do posicionamento de objetos em uma certa estrutura para otimizar o entendimento do que se está querendo transmitir ao usuário e da definição de *templates*, utilizando de um raciocínio que leva em conta o uso, o comportamento e a estética para que as telas sejam atraentes e comuniquem aquilo que querem com facilidade (SCHLATTER; LEVINSON, 2013).

Conceitos básicos do *layout* incluem: alinhamento, proximidade, *grid* e espaço em branco. No contexto de aplicativos também pode-se incluir conceitos como tamanho de telas, *templates* e responsividade. Estes fatores afetam diretamente como o usuário recebe as informações na tela. Alinhamento e proximidade são essenciais para que o usuário consiga entender o que pertence ou está relacionado a o que, botões longe de caixas de texto ou imagens dão ideia de que não estão relacionados, por exemplo. *Grid* é um conjunto de linhas verticais e horizontais invisíveis ao usuário que servem para formar a estrutura do *layout*, providenciando uma área modular para cada objeto, seu alinhamento e seu tamanho. como ilustrado pela Figura 4.

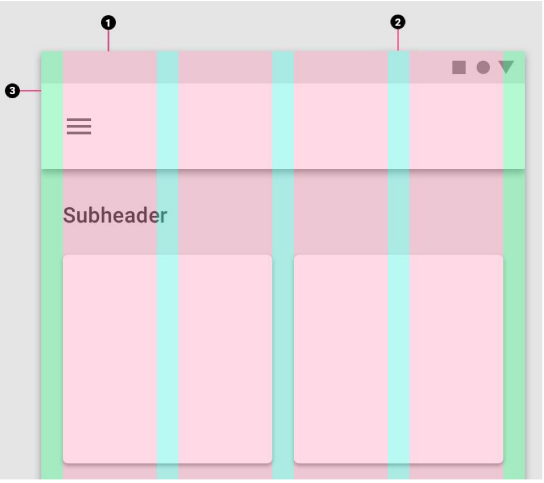

Figura 4: Exemplo de *grid*. (1) Colunas. (2) Espaçamentos. (3) Margens. (Adaptado de Google, 2019a).

O espaço em branco é uma forma de utilizar o espaço para dar foco a determinados objetos e áreas da aplicação.

Um excelente exemplo é a página inicial da Google, que utiliza de bastante espaço em branco para chamar a atenção do usuário para o seu logotipo e caixa de busca, os principais elementos da tela (Figura 5).

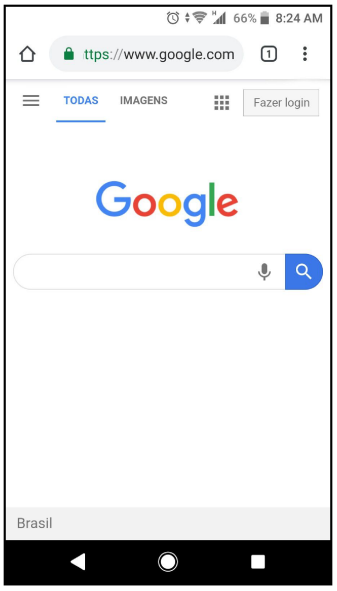

Figura 5: Página inicial da Google (Google, 2019b).

A **tipografia** possui, assim como os outros princípios e "regras", suas próprias convenções. Nascida antes mesmo da imprensa impressa, a tipografia teve seu início na escrita à mão e evoluiu até chegar ao que vemos hoje em questão de fontes para texto em computadores. A palavra "fonte", apesar de usada amplamente para se falar da

aparência de letras em telas de computadores, é na verdade apenas um subconjunto de uma família tipográfica (SCHLATTER; LEVINSON, 2013).

Famílias tipográficas são grupos de letras, números e símbolos relacionados entre si, criados com diferentes (SCHLATTER; LEVINSON, 2013):

- Tipos: serifa, sem-serifa, mono-espaçada, *script*.
- Pesos: níveis de espessura e tamanho.
- Estilos: mudanças na forma e ângulo das letras.alinhamento dentro de cada linha.

Os textos em uma mesma aplicação devem ser consistentes, não exagerados, para que o usuário não tenha dificuldades na sua experiência com a aplicação. Tamanhos do texto, negrito, itálico e outros estilos devem ser usados de acordo com as características desejadas para se criar uma ideia de hierarquia no texto, dando ordem, criando contraste e guiando o usuário pela aplicação.

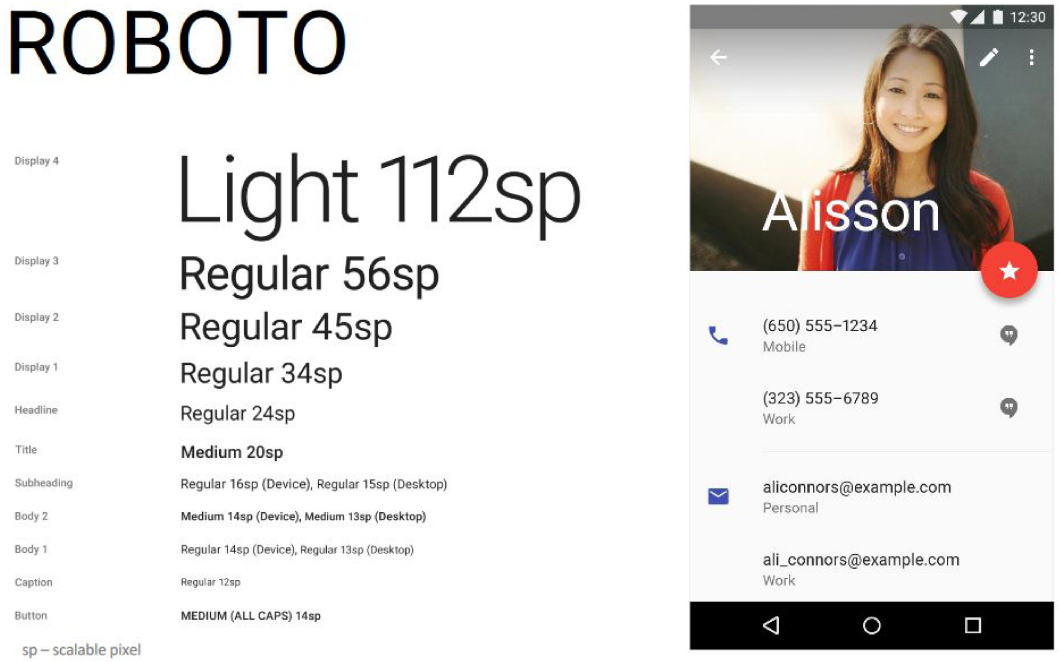

Figura 6: Exemplo da família de fontes sem-serifa Roboto (adaptado a partir de Google, 2019c).

Cor é usada para realçar e destacar, ela ajuda o usuário a saber onde olhar, o que fazer e age como uma espécie de linguagem visual, comunicando ao usuário todo tipo de informação sobre o que está acontecendo na aplicação (SCHLATTER; LEVINSON, 2013). É um elemento chave para atrair, comunicar e guiar o usuário, e de forma geral

apps Android devem utilizar poucas cores, preferivelmente em esquemas com uma cor primária e uma secundária, além de branco, preto e cinza, para evitar sobrecarregar o usuário com muita informação.

As cores primária e secundária, e suas variantes, devem criar um tema harmonioso, que garante texto legível e distingue bem os elementos na interface de usuário (Google, 2019d). Google (2019d) oferece, através de sua linguagem de design *Material Design*, uma palheta de cores “agradáveis” para apps (Figura 7).

Figura 7: Exemplo da palheta de cores do *Material Design* (adaptado a partir de Google, 2019d).

Existem dois modelos comuns de cores relacionados ao design em telas: um geralmente usado para a tintura e outra usada para luz. “Círculos cromáticos” evoluíram com o tempo, porém, todos colocam cores em posições deliberadas no círculo. A versão tradicional deste círculo trabalha com um modelo conhecido como *subtrativo* - da ideia onde inicia-se com uma superfície branca e então utiliza-se da mistura de novas tinturas entre a luz e o olho.

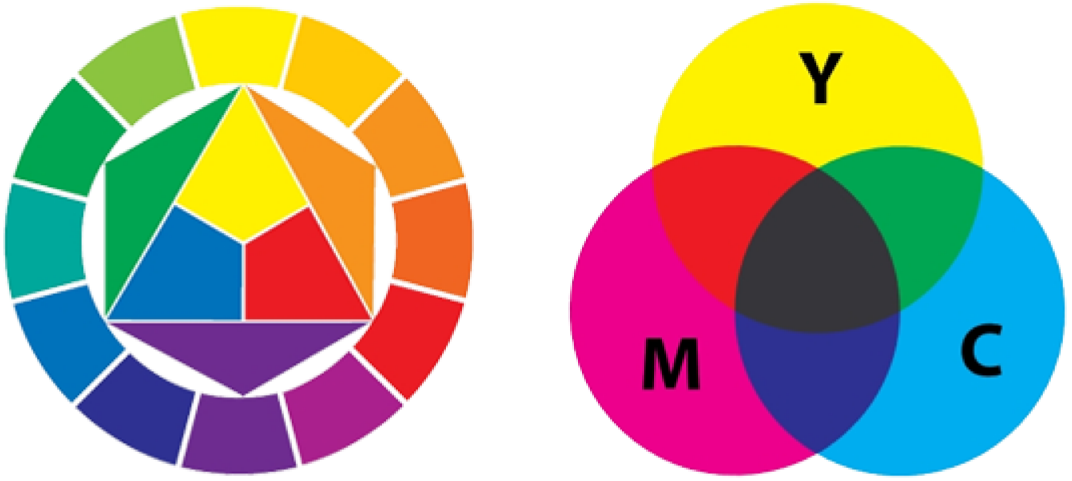

Figura 8: Círculo tradicional à esquerda e exemplo de outro modelo bastante popular à direita (adaptado a partir de ITTEN, 1997).

Cores utilizadas em aplicações precisam de ser planejadas tendo seu objetivo em mente. Após a seleção de cores, decisões sobre a tonalidade e a saturação são tomadas.

Tonalidade são cores saturadas enxergadas nas cores espectrais (vermelho, laranja, amarelo, verde, ciano, azul e violeta) e não-espectrais (magenta, tons de roxo) (BRIGGS, 2019). *Saturação* é a pureza relativa de uma cor em comparação com o cinza (DONDIS, 1973). A Figura 9 ilustra três exemplos de conjuntos de cores que fazem bom uso da saturação para transmitir, cada um, um sentimento diferente.

Figura 9: Diferentes exemplos do uso da saturação em uma palheta de cores.

Contraste, no contexto de cores, se refere a diferença entre duas cores, é a chave para criar hierarquia e atrair o olhar do usuário, além de ajudar a estabelecer personalidade e humor (SCHLATTER; LEVINSON, 2013). Existem outros quatro tipos principais de contraste em cores no design. O contraste quente-frio (Figura 10) - em geral as cores quentes possuem tonalidades de amarelo, vermelho e laranja enquanto tonalidades verde, azul e violeta são cores frias (ITTEN, 1997).

Figura 10: Cores quentes (em cima) e cores frias (embaixo).

O contraste complementar (Figura 11) - utiliza pares de cores opostos um ao outro na roda de cores tradicional para criar uma vibrância e luminosidade. Não significa que este par de cores sejam atraentes quando juntas, mas sim que, quando postas uma do lado da outra com o mesmo valor e saturação, uma causará a outra a ser percebida com uma vibrância maior do que o normal (SCHLATTER; LEVINSON, 2013).

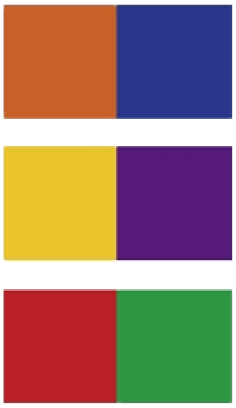

Figura 11: Pares de cores complementares tradicionais.

O contraste por extensão (Figura 12), que parte da ideia em que nem todas as cores são iguais. Por exemplo, em um par amarelo-roxo, amarelo é claro e roxo é escuro, o que nos leva a uma diferença natural em termos de saturação quando apenas retirados diretamente da roda tradicional de cores. Para balancear um tom roxo, apenas um pouco de amarelo é necessário (SCHLATTER; LEVINSON, 2013). Isto levou a criação de uma nova roda, balanceada com o “peso” de cada tonalidade.

Figura 12: Roda de cores tradicional com proporções alterada para balancear o “peso” visual de cada tonalidade (ITTEN, 1997).

No esquema de combinação análoga (Figura 13) se tem um grupo de duas ou três cores que se encontram “ao lado” uma das outras na roda de cores e, portanto, oferecem uma opção de contraste mais sutil.

Figura 13: Exemplo de um conjunto análogo na roda de cores tradicional (adaptado a partir de ITTEN, 1997).

E por último podemos mencionar a combinação triádica (Figura 14) que utiliza de três cores igualmente espaçadas na roda. Esta combinação possui uma boa harmonia e riqueza de cores.

Figura 14: Exemplo de uma combinação triádica na roda de cores tradicional (adaptado a partir de ITTEN, 1997).

Pessoas esperam ver diferentes tipos de **imagens** dependendo da aplicação em que estão usando. Em uma aplicação analítica, há uma expectativa de ver tabelas e gráficos. Em uma aplicação que busca serviços por perto é bom ter um mapa e talvez fotos para facilitar o reconhecimento de lugares. Até mesmo quando usam uma aplicação para instalar outra, pessoas esperam ter imagens de suas telas para ter uma ideia de como esta outra aplicação se parece (SCHLATTER; LEVINSON, 2013).

A Figura 15 ilustra três telas, onde cada uma faz uso de diferentes tipos de imagens: (a) uso de ícones para indicar as ação disponíveis ao usuário, (b) uso de uma imagem relativa ao estabelecimento em questão para melhor identificação e engajamento, bem como o uso de ícones de estrela para indicar a avaliação do estabelecimento, (c) uso de mapa para melhor localização do estabelecimento desejado.

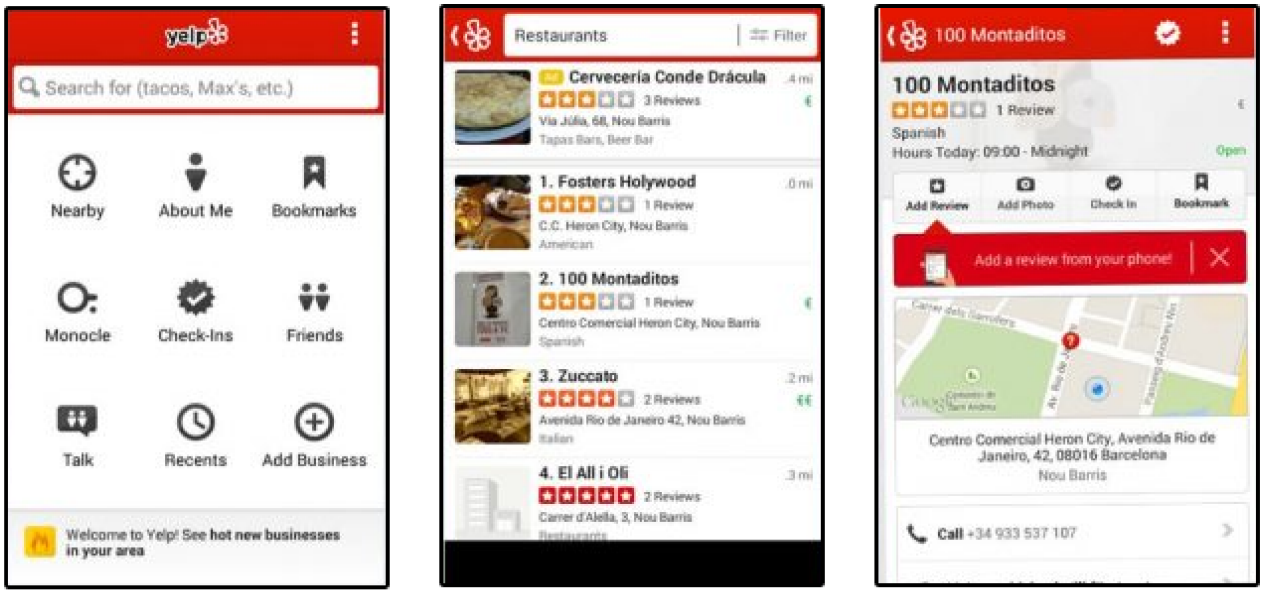

Figura 15: Telas do aplicativo Yelp[4], voltado à avaliação de estabelecimentos comerciais (elaborado pelo autor).

Imagens são uma grande ferramenta para “responder perguntas”, comunicar mensagens de uma maneira diferente, ajudar o usuário a se sentir mais engajado na aplicação.

> “*Use of imagery is a great opportunity to elevate the ordinary and delight people with expression that supports an application’s content.*” (SCHLATTER; LEVINSON, 2013).

Não são apenas um detalhe para deixar a aplicação mais bonita e atraente, elas comunicam informação por meio de três aspectos (SCHLATTER; LEVINSON, 2013):

- Papel: é o motivo pelo qual a imagem foi incluída, ela deve suportar comunicação atraindo a atenção, provendo explicação, evidenciando detalhes, expressando personalidade, convidando interação, ou reforçando padrões.
- Assunto: é o que a imagem retrata, por exemplo o uso de um mapa em um app de localização.
- Qualidades: nitidez, luminosidade, foco, etc.

No contexto de apps, um importante grupo de imagens são os **ícones**. Ícones são imagens simplificadas de alguma coisa, como um objeto ou uma ação. É uma imagem

[4] https://www.yelp.com

que precisa se parecer o suficiente com o que aquilo representa para ser entendido e para ser usado no lugar de uma palavra ou imagem mais detalhada (SCHLATTER; LEVINSON, 2013).

Ícones criados e usados em aplicações geralmente possuem um conjunto, uma família, de ícones de onde se originam. Estes conjuntos seguem um mesmo modelo para que os ícones possuam consistência visual entre si.

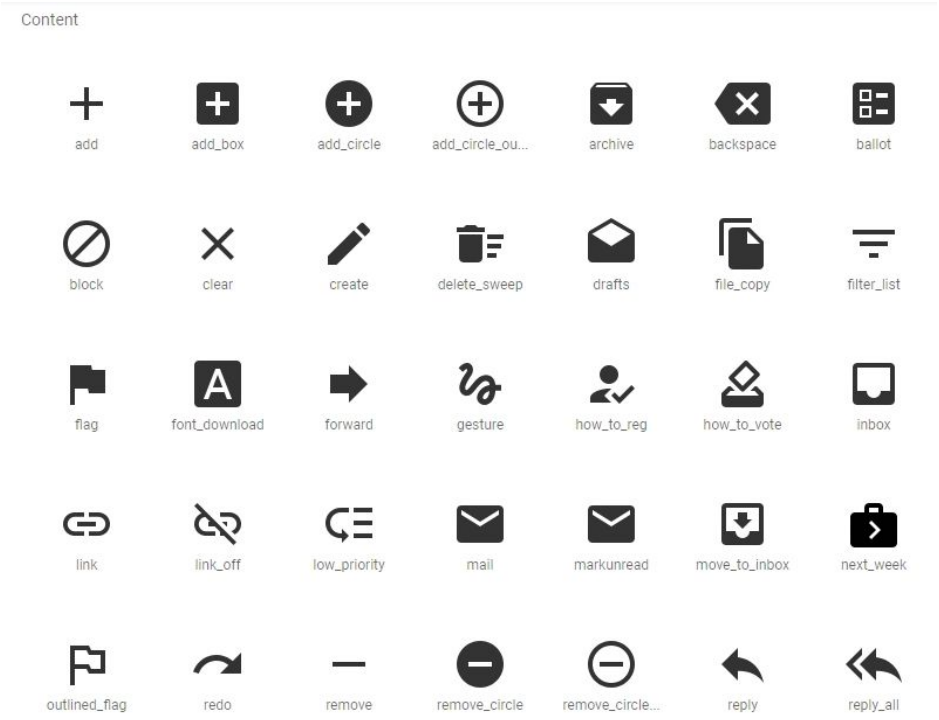

Figura 16: Exemplo de ícones disponibilizados pelo *Material Design* (Google, 2019e).

Ícones desnecessariamente complexos atrapalham mais do que ajudam. O objetivo é transmitir uma mensagem para o usuário de modo rápido, intuitivo, substituindo a necessidade de textos e adicionando um maior interesse visual nas interfaces (SCHLATTER; LEVINSON, 2013).

Todos estes elementos de design visual precisam ser trabalhados em conjunto no desenvolvimento de aplicações. Nosso cérebro funciona primariamente com imagens, por meio do que chamamos de percepção visual[5] (SRINIVASA, 2019). A experiência dos usuários com a aplicação é fortemente afetada por como ele a percebe visualmente. O

[5] Habilidade do cérebro de “dar sentido” às coisas que os olhos enxergam.

estresse visual, que pode ser causado por diversos fatores resultantes da fraca aplicação do design, citando (SRINIVASA, 2019):

- Excesso de dados na tela - podem ser causados por falta de hierarquia, *layout* mal-planejado, uso incorreto de imagens e ícones…
- Dados não parecem organizados - novamente, a falta de hierarquia na apresentação dos dados.
- “Ruído” visual - causado por imagens desnecessárias, contraste ruim entre as cores utilizadas.
- Difícil compreensão de textos - causado tanto por escolhas ruins de cores quanto de tipografia.

Este estresse visual causa rápida fadiga que afeta até mesmo a usabilidade do produto, deixando o usuário confuso e frustrado, muitas vezes até levando-o a abandonar a aplicação por completo. A importância da estética aumenta à medida que a interface entre o artefato e as pessoas afetadas se torna mais abrangente (SRINIVASA, 2019).

2.3 *Deep Learning*

John McCarthy, mundialmente conhecido como um dos pais da Inteligência Artificial (IA), definiu-a como *"a ciência e engenharia de fazer máquinas inteligentes que têm a capacidade de atingir objetivos como humanos"* em 1955 (INDIA, 2018). Logo em seguida, em 1959, Arthur Samuel, pioneiro no campo de estudo de *machine learning*, definiu este como *"um grande subcampo de IA que trata de lidar com o campo de estudo que dá aos computadores a capacidade de aprender sem serem programados explicitamente"* (INDIA, 2018). A Figura 17 ilustra esta definição.

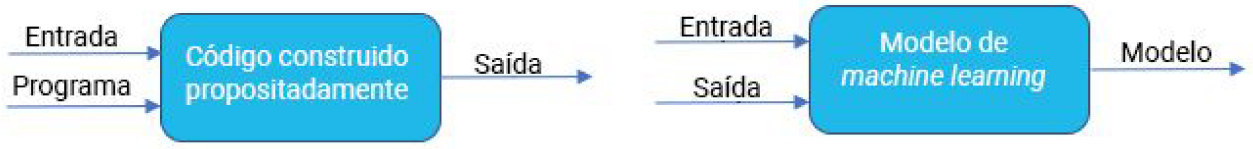

Figura 17: Código construído propositadamente vs modelo *machine learning* (adaptado a partir de INDIA, 2018).

Uma técnica de *machine learning* é o ***Deep Learning***, um tipo específico de aprendizagem de representação utilizado por muitas abordagens da IA (GOODFELLOW et al., 2016). É muito usado para fins de classificação:

- Detecção de faces, identificação de pessoas em imagens, reconhecimento de expressões faciais (raiva, felicidade...);
- Identificação de objetos em imagens (sinais de rua, pedestres…);
- Reconhecimento de gestos em vídeos;
- Reconhecimento de voz, transcrever fala para texto, reconhecimento de sentimentos através da voz;
- Classificar textos como *spam* (em *emails*), ou fraudulentos (em reivindicações de seguros), reconhecimento de sentimentos através da escrita (*feedback* de clientes).

E também para tarefas generativas:

- Criação de imagens fotorealísticas;
- Alteração de expressões faciais em fotos;
- Criação de cenários para jogos de computador;

- Visualização de designs;
- Criação de arte por um computador.

Deep learning alcança grande poder e flexibilidade, pois aprende a representar o mundo como uma hierarquia aninhada de conceitos, com cada conceito definido em relação a outro mais simples, e representações abstratas computadas em termos de outras menos abstratas (GOODFELLOW et al., 2016). A Figura 18 ilustra o relacionamento das diferentes disciplinas de IA, mostrando que *deep learning* é um tipo de *feature learning*, que por sua vez é um tipo de *machine learning*, que é utilizado por várias, mas não todas, as abordagens a IA. E a Figura 19 provém um fluxograma de alto nível sobre como cada uma funciona, evidenciando as similaridades entre seus funcionamentos e como as capacidades de aprendizado a partir de entradas mais simples parecem se tornar maiores na direção de *deep learning*.

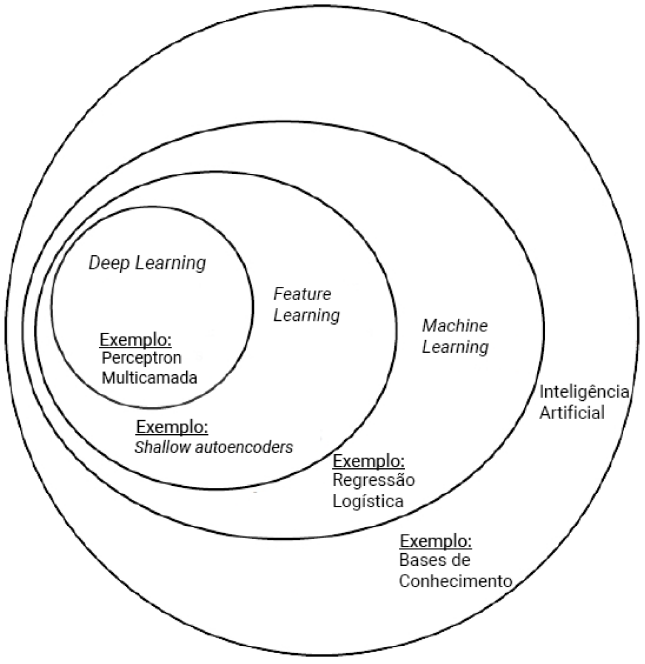

Figura 18: Relação entre *Deep Learning* e a área de inteligência artificial (adaptado a partir de GOODFELLOW et al., 2016).

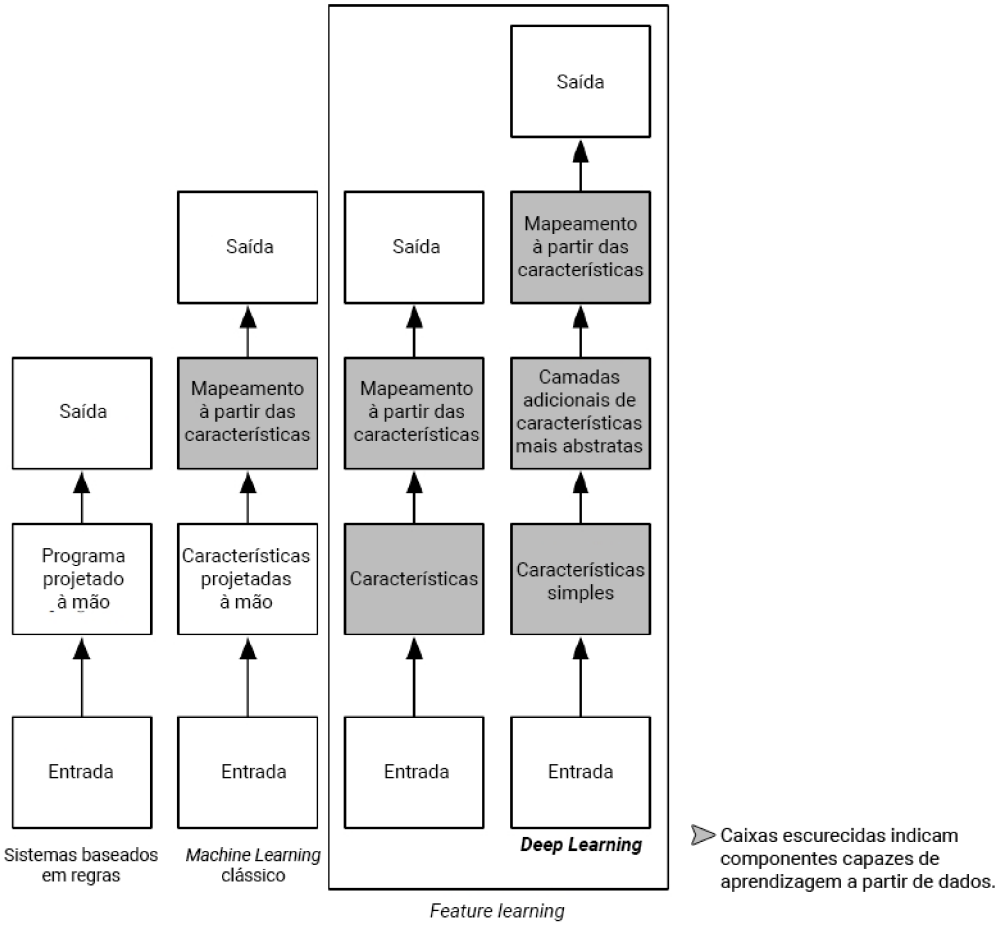

Figura 19: Relação de sistemas de IA em diferentes disciplinas (adaptado a partir de GOODFELLOW et al., 2016).

Deep learning é uma técnica que permite sistemas de computadores se aprimorarem por meio de experiências e dados (GOODFELLOW et al., 2016). *Deep learning* usa de “redes neurais empilhadas”, ou seja, uma rede com múltiplas camadas de redes.

2.3.1 Redes Neurais

As redes neurais são um grupo de algoritmos, vagamente modelados com base no cérebro humano, utilizados para reconhecimento de padrões. Elas interpretam dados sensoriais por meio de uma espécie de percepção de máquina, categorizando ou agrupando entradas puras de dados. Estes padrões reconhecidos por elas são numéricos, contidos em vetores, nos quais todos os dados do mundo real, sejam eles imagens, sons ou textos, precisam ser “traduzidos” (Skymind, 2019a).

Existem diversos diferentes tipos de redes neurais (*perceptron*, *radial basis network*, *deep feed forward*, *recurrent neural network*, *deep convolutional network*...), cada uma projetada para uma determinada tarefa e objetivo, e nem todas tratam da classificação de dados.

2.3.2 Elementos de uma Rede Neural

Uma rede neural é composta por vários nodos conectados entre si por algum tipo de conexão que possuem um peso (numérico) associado a ele (RUSSEL, 2015). Estes pesos são o método primário de memória a longo termo da rede, e o processo de aprendizado geralmente se trata de atualizações destes valores (RUSSEL, 2015). Alguns nodos são conectados ao ambiente externo e são chamados de nodos de entrada ou saída. Cada nodo então faz alguma computação local com a entrada recebida com o que chamamos de função de ativação, que é responsável por ativar ou não um neurônio com base em uma somatória de pesos e viés com o propósito de introduzir uma não-linearidade à saída dos neurônios, característica necessária para que ocorra o aprendizado e a realização de tarefas mais complexas (RUSSEL, 2015).

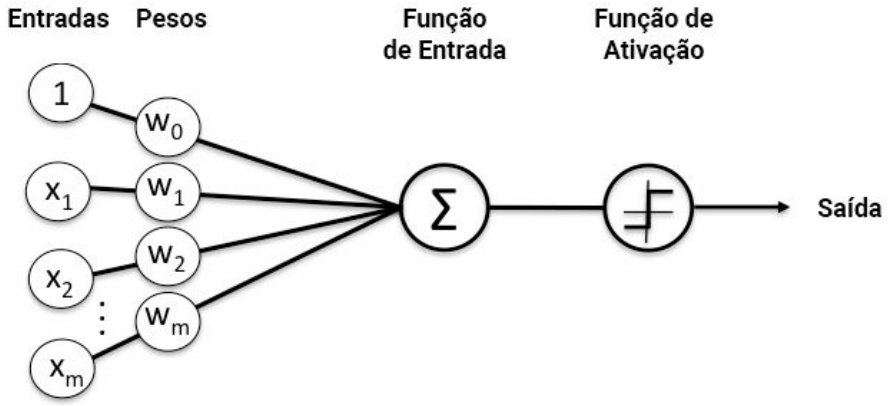

Figura 20: Diagrama exemplo de um nodo (adaptado a partir de Skymind, 2019a).

Uma camada é, então, um conjunto destas unidades que se parecem com neurônios e que “ligam e desligam” à medida que dados são alimentados a eles como entrada e passam pela rede. A saída de cada uma destas camadas é a entrada da camada subsequente, iniciando de uma camada inicial que recebe os dados diretamente do usuário.

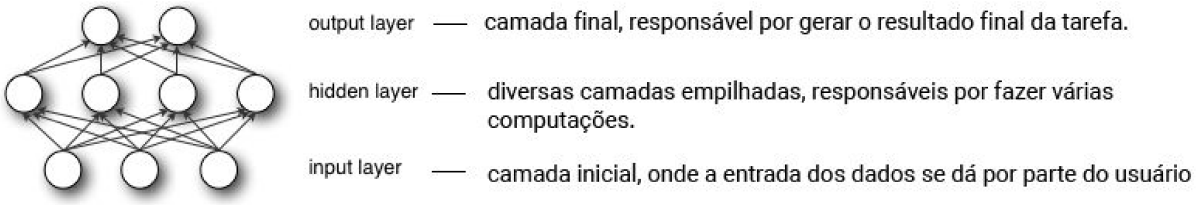

Figura 21: Estrutura geral de uma rede (adaptado a partir de Skymind, 2019a).

2.3.3 *Deep Neural Networks* (DNN)

Existem diversos tipos de redes neurais, uma delas são as *Deep Neural Network* (DNN). As DNNs são uma estrutura de rede usada em *Deep Learning* que se distingue das redes neurais mais comuns por possuir mais de uma camada "escondida" de nodos, pelas quais os dados devem passar e serem submetidos à processos envolvendo múltiplos passos para o reconhecimento de padrões (Skymind, 2019a).

Cada camada da rede treina sob um conjunto distinto de características baseado na saída da camada anterior. Quanto mais se avança na rede neural, mais complexo são as características que os nodos são capazes de reconhecer, pois eles agregam e recombinam características de camadas anteriores, processo conhecido como hierarquia de características (Skymind, 2019a).

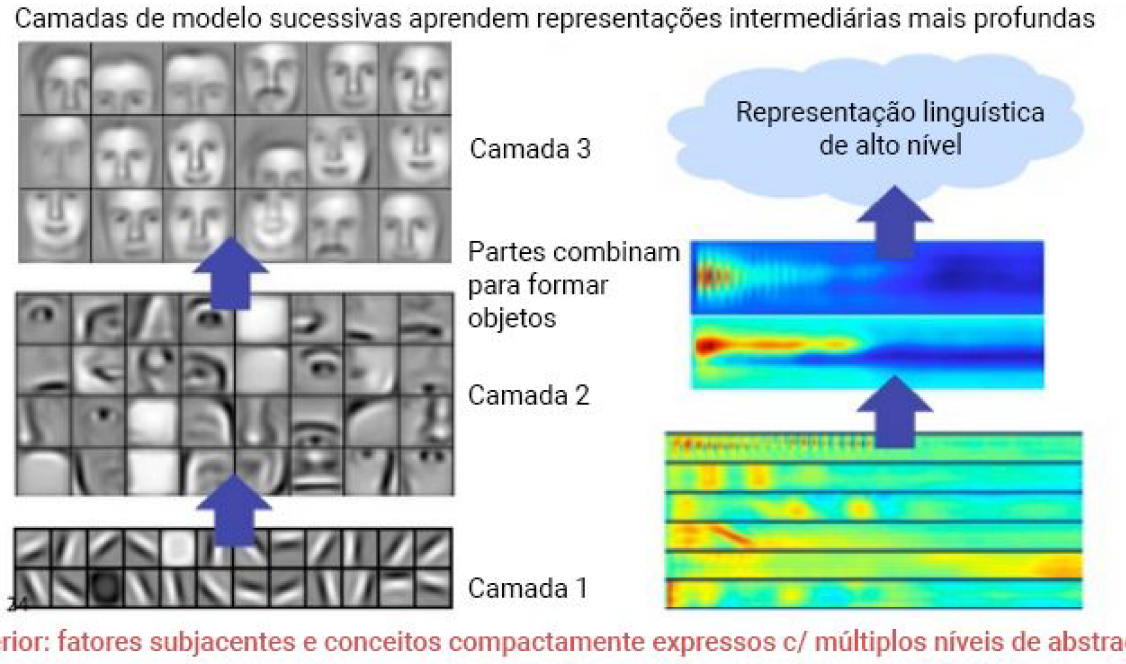

Figura 22: Hierarquia de características; aumenta em complexidade e abstração a cada nível (adaptado a partir de Skymind, 2019a).

Esta característica é o que faz com que *deep neural networks* sejam capazes de lidar com conjuntos de dados muito grandes e de alta dimensão com bilhões de parâmetros que passam por funções não-lineares (Skymind, 2019a). Com isto estas redes são capazes de descobrir estruturas latentes em dados não-estruturados e não-categorizados, que representam a vasta maioria de dados no mundo, também conhecidos como mídia crua; imagens, textos, vídeos e gravações de áudio. Portanto, um dos problemas que *deep learning* melhor resolve é o processamento e agrupamento de mídias, não-categorizadas, discernindo similaridades e anomalias nos dados em que nenhuma pessoa organizou em algum tipo de banco de dados ou até mesmo nomeou anteriormente (Skymind, 2019a).

2.3.4 *Convolutional Neural Networks* (CNN)

Outro tipo de rede neural são as *Convolutional Neural Networks* (CNN).

Este é um tipo de DNN primariamente utilizado para a classificação de imagens (nomear o que a rede enxerga), o agrupamento por similaridade (por exemplo, busca fotográfica), e realizar o reconhecimento de objetos em cenários. São algoritmos capazes de identificar faces, indivíduos, sinais de trânsito, tumores, animais e diversos outros aspectos de dados visuais (Skymind, 2019b).

CNNs podem ser construídas com diversas arquiteturas com diferentes profundidades, diferentes funções de ativação, variação do tipo de cada camada, etc. A Figura 23 apresenta um esquema com diversos modelos tipicamente usados para visão computacional[6].

[6] Sistemas artificiais que obtém informação de imagens ou quaisquer dados multidimensionais.

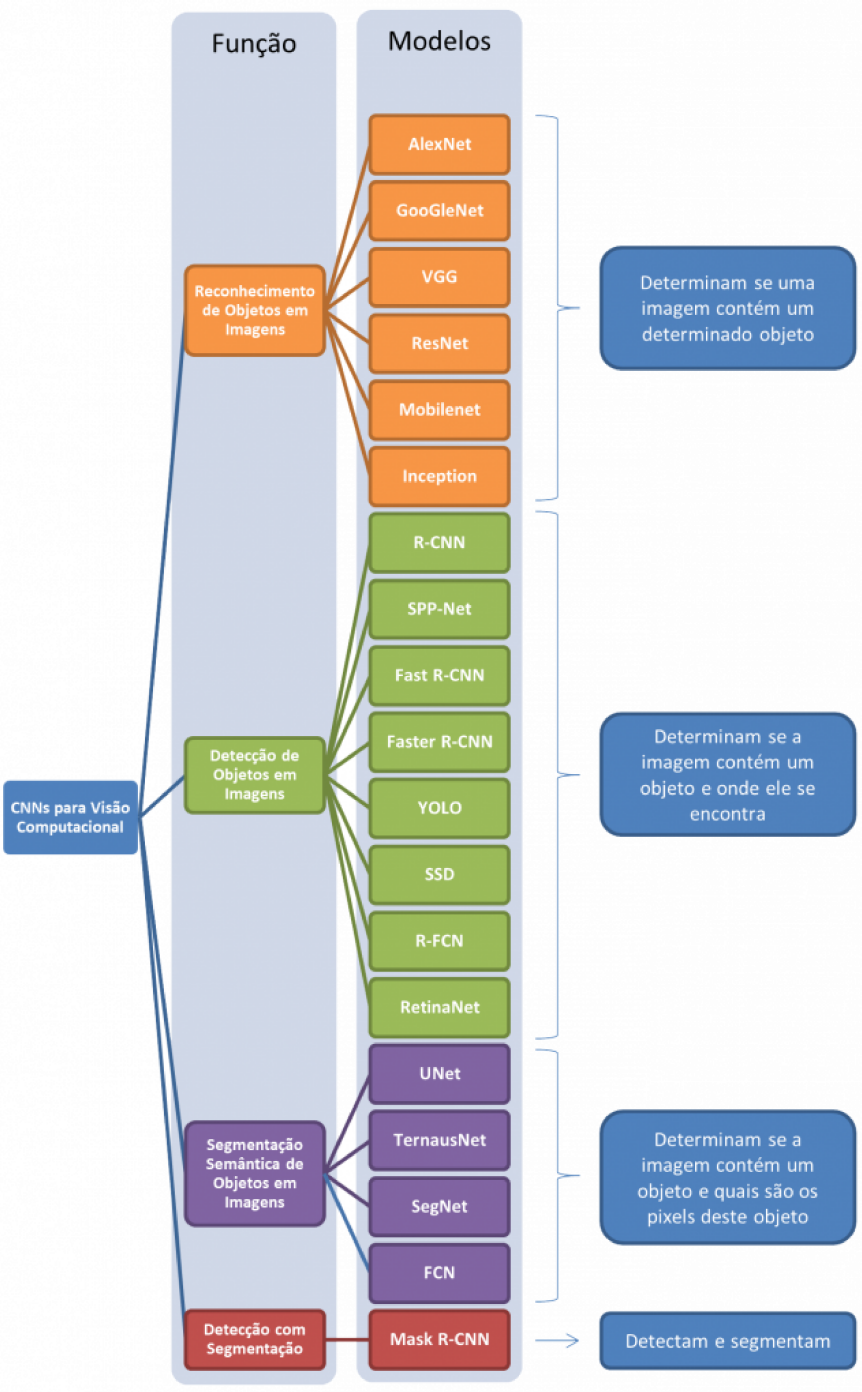

Figura 23: CNNs para visão computacional (VON WANGENHEIM, 2018).

Este tipo de rede neural enxerga imagens como se fossem volumes; objetos tridimensionais. Isto se dá pois uma imagem digital possui uma codificação *Red-Blue-Green* (RGB), que mistura estas três cores para produzir o espectro de cores perceptível pelo humano. Estes objetos tridimensionais são, então, compostos de três

camadas (uma para cada cor) em forma de matrizes, empilhadas umas sobre as outras (Skymind, 2019b).

A rede utiliza destas três matrizes de cores RGB, pegando grupos de pixels de cada vez e os passa por uma outra matriz filtro, realizando uma operação de produto escalar, para então gerar uma nova matriz resultado, chamada de mapa de ativação. Quanto maior o número resultado, mais semelhante o pixel de entrada é em comparação ao padrão expresso pelo filtro. Podemos observar um passo deste funcionamento na Figura 24.

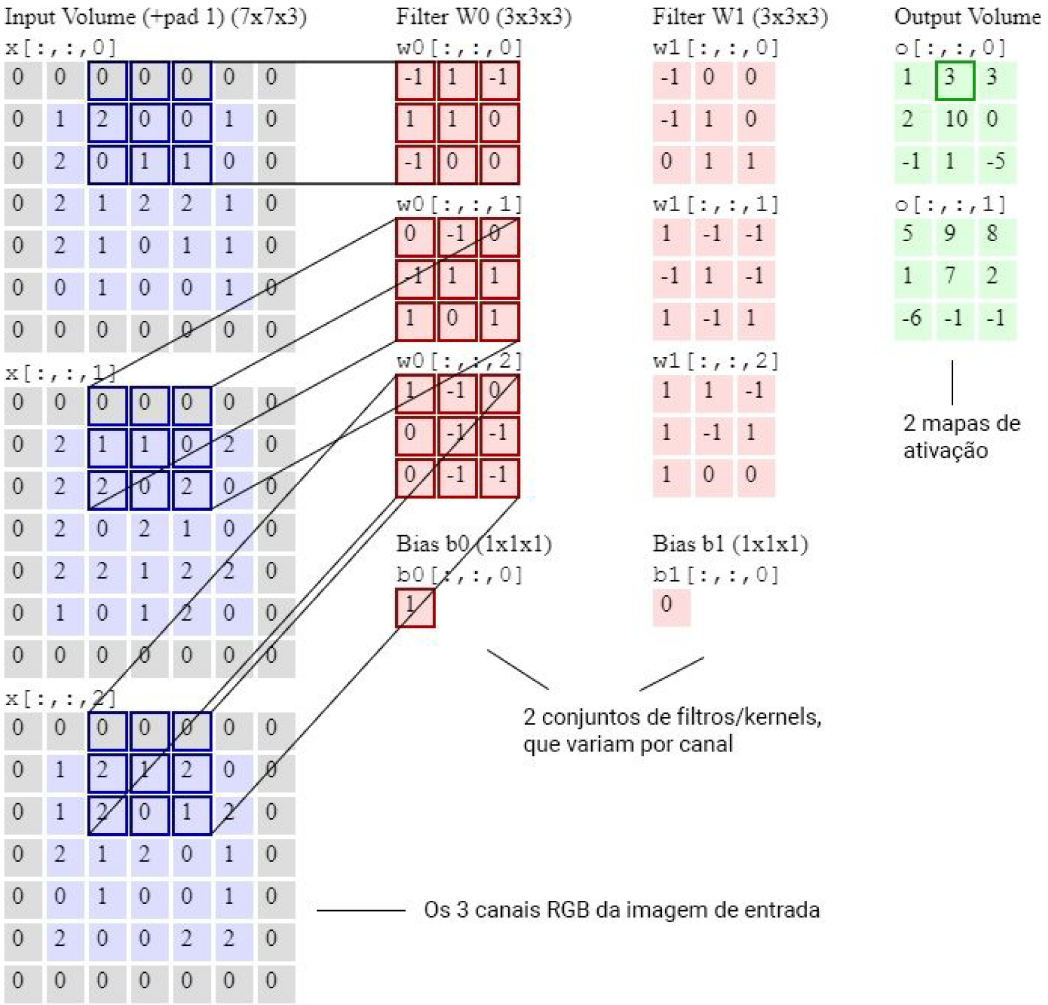

Figura 24: Exemplo com matrizes RGB 30x30 e filtros 3x3 (adaptado a partir de Skymind, 2019b).

Este é um processo de **convolução**, que é, tradicionalmente, uma integral que expressa a sobreposição de uma função *g* ao ser deslocada sobre uma outra função *f*, assim "combinando" uma função com outra (WEISSTEIN, 2019). Convolução pode ser melhor

entendida criando um produto escalar como sendo as duas funções como ilustra a Figura 25.

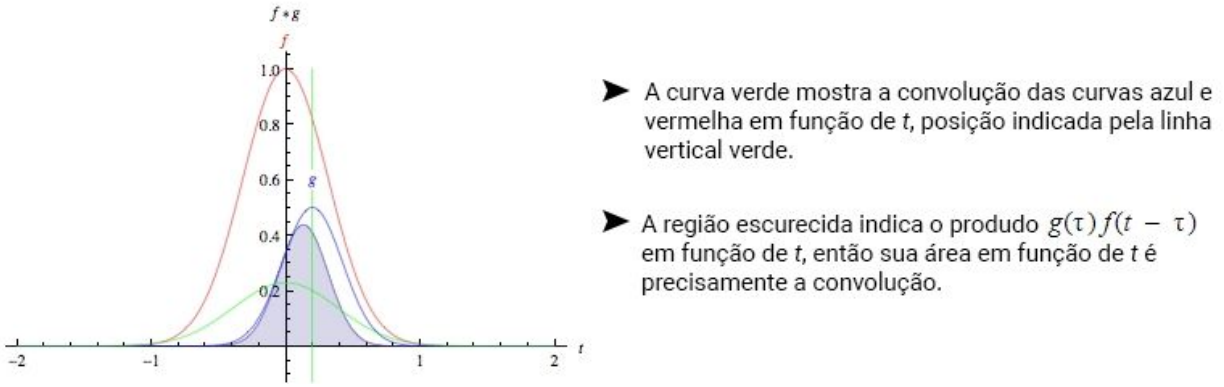

Figura 25: Convolução (adaptado a partir de WEISSTEIN, 2019).

A camada seguinte em uma CNN é conhecida por três nomes: *pooling*, *downsampling* ou *subsampling*; todas significando a mesma coisa. Seu funcionamento se dá da seguinte maneira: os mapas de ativação são alimentados à uma camada de *downsampling*, e assim como em convoluções, este método acontece um trecho de cada vez. Neste caso, *max pooling* simplesmente pega o maior valor de um trecho da imagem e o coloca em uma nova matriz, ao lado do maior valor de outros trechos, e então descarta o restante das informações contidas no mapa de ativação (Skymind, 2019b).

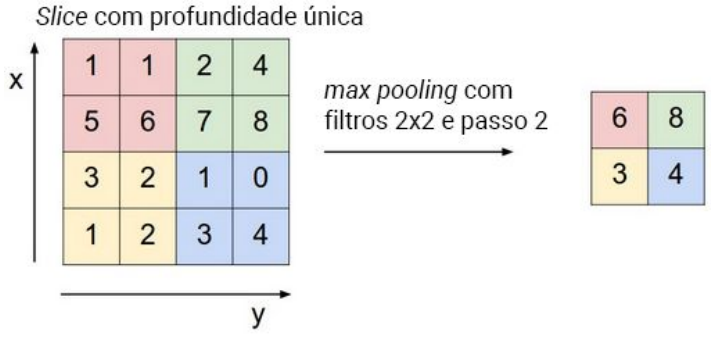

Figura 26: Exemplo do processo de *downsampling* (adaptado a partir de Skymind, 2019b).

A estrutura típica de uma CNN se dá de maneira alternante (Figura 27):

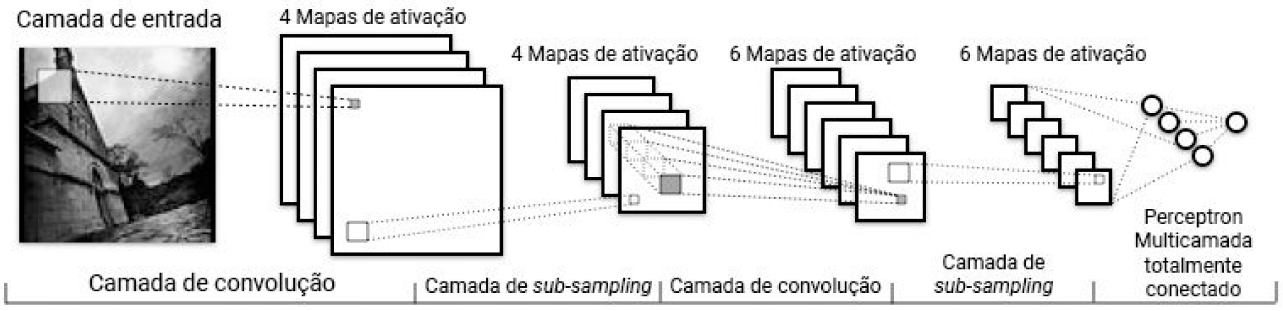

Figura 27: Camadas alternantes em uma CNN típica (Skymind, 2019b).

2.3.5 Treinando uma rede

Focando na **classificação**, foco do presente trabalho, todas as tarefas dependem de um conjunto de dados classificado por um humano para que, então, a rede neural possa começar seu processo de aprendizado por meio da análise e busca de relações entre todos os dados contidos no conjunto de dados. Isto é conhecido como treinamento supervisionado. Outros tipos de treinamentos existentes são:

- Treinamento não-supervisionado: o computador é treinado com dados não pré-classificados por meio de tentativas de formar e/ou encontrar um padrão entre os dados contidos no conjunto de dados. Casos de uso principais são a detecção de padrões e a modelagem descritiva.
- Treinamento semi-supervisionado: uma mescla entre os treinamentos supervisionado e não-supervisionado;
- *Reinforcement learning*: métodos que utilizam de observações e interações com o ambiente na sua tomada de decisão para maximizar o ganho ou minimizar a perda, o algoritmo (chamado de agente) aprende continuamente com suas ações, de maneira iterativa, até cobrir todos os possíveis estados. Casos de uso incluem IAs para jogos de tabuleiro, mãos robóticas e automóveis autônomos (FUMO, 2017).
- *Adversarial learning*: que consiste de duas redes neurais competindo entre si, uma responsável por “criar dados falsos” e a outra responsável por discriminar, entre os dados da primeira e um conjunto de dados passado como entrada, quais dados são reais e quais são falsos (KAHIRI, 2018).

Ao ser criada, os nodos da rede neural recebem ‘pesos’ aleatórios. Estes pesos são atualizados durante o processo de treinamento por meio de algum algoritmo que se está tentando otimizar. Os algoritmos mais usados hoje em dia são o método do gradiente (GD

[7]), o método do gradiente estocástico com *momentum* (SGD *with momentum*), a propagação da raiz quadrática média (RMSprop[8]), o algoritmo Adam e algoritmos genéticos (BUSHAEV, 2017).

O processo de treinamento então consiste na busca de uma função ótima que resulte em uma acurácia maior e satisfatória para a tarefa desejada. Maximização de acurácia é uma tarefa muito difícil, porém, encontrar uma função para minimização de perdas é muito mais fácil e representa o mesmo objetivo (BUSHAEV, 2017).

Por meio do método GD, a rede iniciada com parâmetros aleatórios busca novos valores para cada nodo em busca de diminuir ao máximo a função de perda e repete este processo diversas vezes até encontrar o mínimo valor possível (TSENG et al, 2019). Para descobrir estes novos valores é necessário calcular o gradiente da função de custo. O gradiente é uma generalização de uma derivada: é um vetor contendo cada uma das derivadas parciais da função com respeito a cada variável, em outras palavras é um vetor que contém a inclinação da função de perda ao longo de cada eixo (TSENG et al, 2019).

Após determinar a direção a ser tomada é necessário decidir o tamanho do passo a ser tomado. Esta consideração, em modelos ordinários de GD, é deixada como um hiperparâmetro: parâmetros a serem decididos e definidos manualmente, antes de realizar o processo de treinamento. Este hiperparâmetro em específico é conhecido como taxa de aprendizagem e é considerado o parâmetro mais importante e sensível a ser determinado (TSENG et al, 2019). Após determinado o valor do taxa de aprendizagem o treinamento é repetido diversas vezes, atualizando os parâmetros de cada nodo a cada nova iteração até que resultados satisfatórios são alcançados. Contudo, alguns cuidados precisam ser tomados em relação ao conjunto de dados a ser usado no processo de treino.

Overfitting descreve uma situação em que a rede foi "super otimizada" e se tornou capaz de "memorizar" o conjunto de dados de treino, ao custo de generalizar o aprendizado para dados desconhecidos (que é o objetivo do treinamento) (TSENG et al, 2019). A Figura 28

[7] do inglês Gradient Descent
[8] do inglês Root Mean Square propagation

ilustra um exemplo visual de *overfitting*, com 11 pontos de dados (pontos pretos) e duas funções são treinadas para “encaixá-los”. A primeira, em linha reta, é simples e captura os pontos com alguma margem de erro. A segunda é extremamente curva e complexa, apesar de apresentar erro zero (captura todos os 11 pontos de dados) ela não generaliza bem.

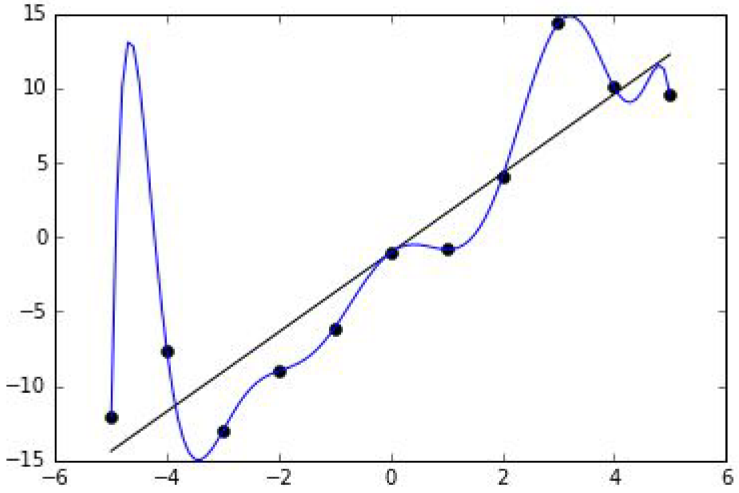

Figura 28: Exemplo de *overfitting* (WIKIPEDIA, 2016).

Para evitar o *overfitting* a solução mais simples é de dividir o conjunto de dados em um conjunto de treino, um conjunto de testes e um outro conjunto de validação; como o grupo de dados a ser usado nos testes e na validação não estava presente no conjunto de treino a rede não é capaz de memorizá-lo (TSENG et al, 2019). Geralmente é utilizado uma separação de 70 a 80% do conjunto de dados para treino e o restante igualmente dividido entre os conjuntos remanescentes (TSENG et al, 2019). A rede é então treinada no conjunto de treino e avaliada no conjunto de validação para se obter os valores ótimos para os hiperparâmetros e descobrir quando parar de treinar (tipicamente quando valores de acurácia no conjunto de validação param de melhorar) (TSENG et al, 2019).

2.4 *Frameworks* e bibliotecas de *Deep Learning*

Tanto *frameworks* quanto bibliotecas são códigos escritos por uma pessoa e que são usados por terceiros para resolver problemas comuns, ou seja, trata-se de código reusável. Seu propósito é facilitar a resolução de problemas comuns. A diferença entre estes dois está na inversão de controle. Quando o desenvolvedor chama um método de uma biblioteca, ele está no controle, já com um *framework*, o controle é invertido, é o *framework* que chama o desenvolvedor (FOWLER, 2005). O *framework* já possui todo o

fluxo de controle e vários “espaços em branco” pré-definidos que é onde o desenvolvedor precisa preencher com seu código. A Tabela 2 traz uma lista dos *frameworks* predominantes hoje, mundialmente.

Tabela 2: *Frameworks* predominantes mundialmente (elaborado pelo autor)

Framework	**Desenvolvido por**	**Principais linguagens compatíveis**
Tensorflow	Google Brain Team	C++, Python
The Microsoft Cognitive Toolkit (CNTK)	Microsoft	C++, C#, Python, Java
Torch	Ronan Collobert, Koray Kavukcuoglu, Clement Farabet	C, LuaJIT
PyTorch	Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan	C++, Python
Keras	François Chollet	Python
Caffe	*Berkeley AI Research* (BAIR)	C++, Python
Eclipse Deeplearning4j	Skymind	Java, Scala, Clojure, Kotlin
Chainer	Preferred Network em parceria com IBM, Intel, Microsoft e Nvidia	Python

Cada um desses *frameworks* possui suas vantagens e desvantagens, tornando-os todos opções viáveis dependendo de sua aplicação. Contudo, *frameworks* requerem um esforço maior por parte do usuário para entender e fazer uso de suas funções quando comparados à bibliotecas. Portanto, com base nesses *frameworks* foram desenvolvidas bibliotecas visando aumentar o nível de abstração e assim facilitar o seu uso e tornar o poder do estado-da-arte em *deep learning* mais acessível e disponível para qualquer pessoa (Fastai, 2019).

Uma das bibliotecas predominantes é a biblioteca **fast.ai**. Ela utiliza do *framework* PyTorch v1 e provê uma única e consistente *Application Programming Interface* (API) para as aplicações de *deep learning* e tipos de dados mais importantes (HOWARD, 2018). Adicionalmente, essa biblioteca fornece estratégias avançadas de otimização de hiperparâmetros fáceis de se usar, que ajudam não só na implementação mas também nos treinos da rede neural (HOWARD, 2018). A estrutura utilizada pela fast.ai, PyTorch, foi considerada uma das estruturas de CNNs com o melhor desempenho, flexibilidade e orientação à pesquisa disponível (FONNEGRA; BLAIR; DÍAZ, 2017). Pelo presente

trabalho ser focado em pesquisas sobre design de interfaces usando técnicas de *deep learning*, em vez de pesquisas sobre *deep learning*, a biblioteca fast.ai apresenta boas condições para a produção de código simples, rápido e eficaz no desenvolvimento deste trabalho.

3. ESTADO DA ARTE

Neste capítulo é apresentado o estado da arte atual de pesquisas e desenvolvimentos relacionados a modelos de avaliação da estética visual de interfaces de usuários de aplicativos usando técnicas de *deep learning*. O estado da arte foi levantado realizando um mapeamento sistemático da literatura seguindo o processo proposto por Petersen et al. (2008).

3.1 Definição do protocolo do mapeamento

O objetivo desse mapeamento é de responder a seguinte pergunta de pesquisa: quais modelos existem para a **avaliação automatizada da estética visual de interfaces de usuários de aplicativos usando técnicas de *deep learning***?

Esta questão é refinada nas seguintes perguntas de análise:

PA1. Quais abordagens existem?

PA2. Quais arquiteturas de redes foram utilizadas?

PA3. Quais os conjuntos de dados utilizados para treinamento e seus tamanhos?

PA4. Qual algoritmo de aprendizado foi utilizado?

PA5. Qual a precisão/qualidade dos resultados obtidos?

Como base são considerados SCOPUS, ScienceDirect, IEEE Xplore Digital Library, ACM Digital Library, Elsevier e Springer. São considerados somente artigos acessíveis via o Portal CAPES[9]. Pela adesão da comunidade para acelerar a divulgação de resultados é considerado também o arXiv.

De acordo com a pergunta da pesquisa são definidos os termos de busca, sinônimos e traduções conforme a Tabela 3.

[9] Um portal da internet para acesso ao conhecimento científico produzido internacionalmente, gerido pelo Ministério da Educação (MEC) do Brasil e direcionado a instituições autorizadas, incluindo universidades, agências governamentais e empresas privadas (www.periodicos.capes.gov.br).

Tabela 3: Termos de busca e sinônimos (elaborado pelo autor)

Termo	**Sinônimos**	**Tradução (inglês)**
Estética	Estética visual, design	*Aesthetics, visual aesthetics,* design
Interfaces de usuário	-	*User interface, UI*
Aplicativos móveis	App, Android	*App, android, mobile application*
Deep learning	Inteligência artificial, avaliação automática, redes neurais	*Deep learning, artificial intelligence, neural networks*

Após a definição dos termos e seus sinônimos, definiu-se o *string* de busca padrão a ser aplicado nas bases de dados:

aesthetics AND ("user interface" OR ui OR web OR websites OR app OR android) AND ("deep learning" OR "artificial intelligence" OR "neural networks")

Os critérios para inclusão e exclusão de artigos encontrados são:

- São considerados somente artigos e artefatos em que se fala sobre estética visual;
- São considerados somente artigos e artefatos que abrangem o uso de inteligência artificial;
- São incluídos apenas artefatos em inglês ou português;
- São incluídos apenas artefatos acessíveis pela CAPES;
- Com o intuito de tornar esta pesquisa mais ampla, são incluídos resultados que tratam de qualquer tipo de interface de usuário de software incluindo *websites*;
- Não são incluídos artigos que contenham fotografia como foco do estudo.

Critérios de qualidade: Consideramos apenas artigos que apresentam informações substanciais sobre a avaliação da estética visual de interfaces de usuários usando técnicas de *deep learning*.

3.2 Execução da busca

A busca dos artigos foi realizada em junho de 2019 pelo autor do presente trabalho e revisada pela orientadora. A busca inicial resultou em 6524 artigos.

Tabela 4: Resultados da busca (elaborado pelo autor)

Base de dados	Quantidade de artigos resultantes da busca	Quantidade de artigos analisados	1ª etapa	2ª etapa
SCOPUS	39	39	1	1
ScienceDirect	566	566	2	1
IEEE	15	15	2	0
ACM	59	59	2	0
SpringerLink	5878	1488	3	0
arXiv	6	6	1	0
Total	**6524**	**2134**	**13**	**2**

A base de dados SpringerLink apresentou diversos artigos abrangendo áreas além do escopo deste trabalho e, por isto, nesta base, foram analisados apenas os artigos da disciplina (*subject)* Ciência da Computação nas subdisciplinas *Artificial Intelligence (incl. Robotics)* e *User Interfaces and Human Computer Interaction*. Nas demais bases todos os resultados foram analisados.

A partir do grupo inicial de artigos para análise, na primeira etapa foram selecionados artigos potencialmente relevantes de acordo com os critérios de inclusão e exclusão por meio de uma rápida análise do título, resumo e palavras-chave de cada artigo. Esta etapa resultou em 13 artigos potencialmente relevantes.

Durante a primeira etapa, vários artigos foram descartados por se tratarem de avaliações estéticas “manuais”, sem o envolvimento de IA. Outra grande parte possuía foco na relação *User Interface vs User Experience* e foram descartados por estarem fora do nosso escopo. Alguns apresentavam estudos nas áreas de segurança (identificação facial, reconhecimento de voz, etc.), outros tratavam de sistemas de recomendações (imagens, fotos, vídeos, filmes, etc.). Vários artigos também foram descartados aqui por tratarem da criação de interfaces de usuário com o auxílio de IAs, e não da sua avaliação.

Na segunda etapa, por terem sido encontrados poucos artigos relevantes, os mesmos foram lidos integralmente. Foram excluídos artigos por mostrarem um foco grande apenas em imagens, como fotografias (APOSTOLIDIS et al., 2019 e KAIRANBAY et al., 2016),

não considerando aspectos da interface, e também por focar em casos de uso e aplicações desta tecnologia. Existe também trabalho voltado à ícones (HOU et al., 2013) mas que não utilizaram de IA no seu processo de análise e, por isso, foram descartados. Esta etapa resultou em 2 artigos potencialmente relevantes.

3.3 Análise dos resultados

Para responder à questão de pesquisa, extraímos as informações relevantes de cada artigo, as quais são apresentadas a seguir.

PA1. Quais abordagens existem?

Observa-se, mesmo com amplas discussões e trabalhos já sendo vistos para imagens e fotografias, quase não existe pesquisa nessa área com foco em interfaces de software. Somente 2 artigos foram encontrados com foco para interface de usuários, ambos voltados a *websites* (Tabela 5). Nenhum estudo publicado com foco em aplicativos móveis foi encontrado.

Tabela 5: Artigos (elaborado pelo autor)

Citação	Referência Bibliográfica	Objeto da análise	Elementos de interface analisado
(KHANI et al., 2016)	KHANI, M.G.; MAZINANI, M.R.; FAYYAZ, M.; HOSEINI, M.; **A novel approach for website aesthetic evaluation based on convolutional neural networks**. Proceedings of the Second International Conference on Web Research, Tehran, Iran, 2016. p. 48–53.	*Websites*	Geral (visual)
(DOU et al., 2019)	DOU, Q.; ZHENG, X. S.; SUN, T.; HENG, P.; **Webthetics: Quantifying webpage aesthetics with deep learning**. International Journal of Human-Computer Studies, vol. 124, 2019. p. 56–66.	*Websites*	Geral (visual)

Ambos os artigos sobre *websites* abordam a questão da estética de forma geral sobre a interface (página), não tratando de elementos em específico.

PA2. Quais arquiteturas foram utilizadas?

Os trabalhos utilizaram de arquiteturas um pouco diferentes, mas possuem uma estrutura de camadas bastante similar (Tabela 6). Ambos usaram de 5 camadas de convolução e 2 MLPs totalmente conectados, porém, DOU et al (2019) fez uso de uma camada a mais (3) para *max pooling* e uma camada extra na saída para regressão.

A Figura 29 mostra a arquitetura de KHANI et al (2016), com camada de entrada 227x227, seguido de uma camada de convolução com *kernel* 11x11 e passo (*stride*) 4 que resulta em 96 mapas de tamanho 55x55. Uma camada de *max pooling* para reduzir o impacto de rotações presentes no dado de entrada, seguida de uma segunda convolução com *kernel* 5x5 e passo 1, gerando 256 mapas de tamanho 27x27. Após mais 3 camadas de convolução há uma última camada de *max pooling*, e então as duas camadas totalmente conectadas (MLPs).

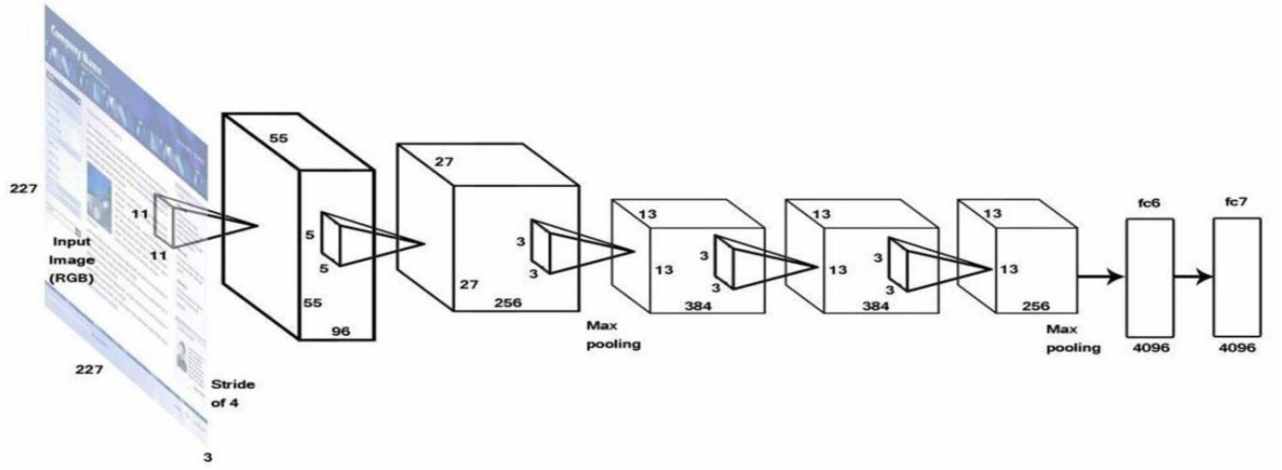

Figura 29: Arquitetura da rede de KHANI et al. (KHANI et al., 2016)

DOU et al. (2019) optaram por reduzir a quantidade de neurônios nas suas duas camadas de MLPs para reduzir as chances de *overfitting* e alteraram a camada final, que continha neurônios treinados no conjunto de dados ImageNet por uma camada de um único neurônio (*single-neuron layer*) para realizar a regressão que trata de estimar um valor de avaliação à cada *input* (página da *web*), ao invés de atribuir um simples *label*.

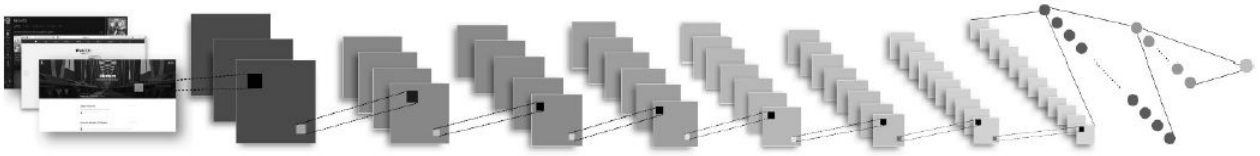

Figura 30: Arquitetura da rede de DOU et al. (DOU et al., 2019)

Layer	Kernel	Stride	Channel
conv1	11 × 11	4	96
pool1	3 × 3	2	96
conv2	5 × 5	1	256
pool2	3 × 3	2	256
conv3	3 × 3	1	384
conv4	3 × 3	1	384
conv5	3 × 3	1	256
pool5	3 × 3	2	256
fc6	–	–	1024
fc7	–	–	512
Regression	–	–	1

Figura 31: Detalhamento das camadas escondidas de DOU et al. (DOU et al., 2019)

Tabela 6: Arquiteturas utilizadas em cada artigo (elaborado pelo autor)

Citação	***Framework***	**Pré-treinamento (conjuntos de dados)**	**Tipo da rede**	**Camadas escondidas**	**Camada de saída**
(KHANI et al., 2016)	Baseada em AlexNet	Places; ILSVRC 2012; LaMem (*fine-tuning*);	CNN	5 camadas de convolução, 3 de *max pooling*.	2 MLPs totalmente conectados.
(DOU et al., 2019)	Baseada em CaffeNet	Flickr-Style; ImageNet;	CNN	5 camadas de convolução, 2 de *max pooling*.	2 MLPs totalmente conectados e 1 de regressão.

PA3. Quais os conjuntos de dados utilizados para treinamento e seus tamanhos?

Ambos trabalhos usaram de redes pré-treinadas (Tabela 6). Este pré-treino preparou as redes para reconhecer e avaliar imagens e fotografias através de grandes conjuntos de dados públicos:

- Places: banco de imagens fotográficas de locais reais, categorizadas e preparadas para uso em redes neurais (ZHOU et al., 2018).
- ILSVRC 2012: banco de imagens ImageNet do ano de 2012 para uso em redes neurais para treino de reconhecimento (RUSSAKOVSKY, et al., 2015).
- LaMem: banco de imagens voltado à memorabilidade (KHOSLA et al., 2015).
- Flickr-Style: banco de imagens voltado ao treino para o reconhecimento artístico de fotografias (KARAYEV, 2019).
- ImageNet: enorme banco de imagens para diferentes propósitos, mas principalmente usado no treino de redes para o reconhecimento de objetos/figuras (ImageNet, 2019).

Para treinar a rede para o propósito de classificação da estética visual, os trabalhos

usaram conjuntos de dados conforme apresentado na Tabela 7. Interessante notar que ambos KHANI et al. (2016) e DOU et al. (2019) utilizaram um conjunto de dados extremamente pequeno de *screenshots* de *websites* neste passo.

Tabela 7: Conjuntos de dados utilizados no treino da rede em cada artigo (elaborado pelo autor)

Citação	Tamanho do(s) conjunto(s) de dados	Resolução das imagens do(s) conjunto(s) de dados	Conjunto(s) de dados base	Separação do(s) conjunto(s) de dados	Tipo do *labeling*
(KHANI et al., 2016)	418 *screenshots* de *websites* com avaliação online de diversos* voluntários	1024x768, *downsampled* para 227x227	Reinecke *and* Gajos (2014) (REINECKE et al., 2014)	376 imagens (90%) para treino; 42 imagens (10%) para teste; **	Escala Likert [1, 9]
(DOU et al., 2019)	398 *screenshots* de *websites* com avaliação de aprox. 40.000 usuários online	1024x768, *downsampled* 4 vezes para 256x192	Reinecke *and* Gajos (2014) (REINECKE et al., 2014)	300 imagens (75.4%) para treino; 98 imagens (24.6%) para teste; Separação realizada de maneira aleatória	Escala Likert [1, 9]

*Número exato não informado

**Não mencionam de que forma foi realizada a separação

Ambos estudos mantiveram a classificação (*labeling*) das imagens utilizadas no conjunto de dados de Reinecke *and* Gajos (2014), a qual foi realizada de maneira online por meio de um estudo de usuário[10]. Neste estudo, usuários eram apresentados uma imagem por vez, por 500 milissegundos, e então avaliavam a imagem em três pontos: complexidade, *colorfulness* e o apelo visual, antes de prosseguir para a próxima imagem. Cada usuário avaliou 30 imagens aleatórias duas vezes, para melhorar a acurácia das avaliações.

Ambos estudos realizaram um pré-processamento nas imagens do conjunto de dados. KHANI et al. (2016) reduziram as dimensões das imagens (*downsampling*) para 227x227. Enquanto DOU et al. (2019) optaram por uma redução para 256x192.

PA4. Qual algoritmo de aprendizado foi utilizado?

KHANI et al. (2016) utilizaram de um SVM (*Support Vector Machine*) com uma função de base radial Gaussiana (*Gaussian Radial Basis Function, Gaussian RBF* (CORTES, 1995)) para a tarefa de classificação e não mencionaram qual algoritmo utilizaram no

[10] LabintheWild.org

treinamento de sua rede neural.

DOU et al. (2019) utilizaram o algoritmo clássico de *backpropagation* com taxa de aprendizado inicial de 0.001 o qual foi recalibrado ao longo do processo de treino, sendo reduzido por um fator de 10 a cada 2 K iterações. Cada iteração possuía um carga de 64 imagens; o momentum foi definido em 0.9; decaimento do peso (λ) definido em 0.001. Utilizaram também a estratégia de *dropout* com taxa de 0.5 para evitar *overfitting* e melhorar a capacidade de generalização das camadas de saída.

PA5. Qual a precisão/qualidade dos resultados obtidos?

Tanto KHANI et al. (2016) quanto DOU et al. (2019) relatam o desempenho de seus trabalhos por meio da taxa de erros (porcentagem da quantidade de predições erradas) observada por seus modelos.

Tabela 8: Precisão/taxa de erros relatada em cada artigo (elaborado pelo autor)

Citação	Taxa de erros
(KHANI et al., 2016)	A taxa de erros reportada pelos autores foi de 34.15%
(DOU et al., 2019)	A taxa de erros reportada pelos autores foi de 20.41%

Apesar das taxas de erros divulgadas ainda parecerem altas, elas são aceitáveis por se tratarem de trabalhos inéditos e servem como um ponto de referência. DOU et al. (2019) sugerem que a melhor forma de tratar o problema é tratá-lo como um problema de regressão, e não de classificação, justificando que predições “mais longes da verdade” deveriam resultar em maiores perdas. DOU et al. (2019) atribuem as melhoras de performance obtidas em relação à KHANI et al. (2016) principalmente ao modo de treinamento utilizado e a alteração do problema para um de regressão.

3.4 Discussão

Apesar do enfoque principal na avaliação nas áreas de fotografias e imagens gerais, pesquisas na área de estudo sobre a avaliação da estética visual de forma geral vem crescendo.

Porém, o levantamento do estado da arte mostrou que, até então, ainda não existem trabalhos explicitamente voltados ao desenvolvimento de um modelo para avaliação da

estética visual de aplicativos móveis com o uso de técnicas de *deep learning*. De forma geral, ainda se observa pouca pesquisa na área de interfaces de usuários de software, vendo poucos trabalhos interessantes surgindo somente voltados à avaliação de *websites*.

Os dois trabalhos encontrados referente a avaliação da estética visual de *websites,* KHANI et al. (2016) e DOU et al. (2019), falam do poder de extração de características que as redes CNN possuem e como demonstram bom desempenho na avaliação de *screenshots* de *websites* com respeito a estética visual dos mesmos. Ambos utilizaram de arquiteturas similares e o mesmo conjunto de dados como base em seus trabalhos, porém, DOU et al. (2019) abordou o problema como um de caráter de regressão e atribuiu as melhores de performance, principalmente, a esta decisão.

Ameaças à validade da revisão da literatura. Como em qualquer mapeamento sistemático, existem ameaças à validade dos resultados encontrados. Estratégias de mitigação para minimizar os impactos das ameaças identificadas foram aplicadas:

- **Viés de publicação**: mapeamentos sistemáticos podem sofrer do viés comum de que os resultados positivos têm maior probabilidade de serem publicados do que os negativos. No entanto, consideramos que os resultados dos artigos, sejam positivos ou negativos, têm apenas uma pequena influência sobre esse mapeamento sistemático, uma vez que foi buscado identificar os modelos existentes para avaliação da estética visual com uso de *deep learning*.
- **Identificação de estudos**: a omissão de estudos relevantes é outro risco a ser considerado. A fim de mitigá-lo, foi construída cuidadosamente a *string* de busca para ser o mais abrangente possível, considerando não apenas os principais conceitos, mas também sinônimos e foram consideradas várias bases de dados de renome.
- **Seleção e extração de dados de estudos**: Ameaças para estudar seleção e extração de dados foram mitigadas por meio do fornecimento de uma definição detalhada dos critérios de inclusão e exclusão e de qualidade. Um protocolo foi rigidamente definido e documentado para a seleção do estudo.

4. DESENVOLVIMENTO DO MODELO “Appsthetics”

Neste capítulo é apresentado o modelo desenvolvido neste trabalho, nomeado “Appsthetics”.

4.1 Análise de Requisitos

Adotando a notação de descrição de Mitchell (1997), o objetivo é desenvolver um programa de computador que aprenda com a experiência E com relação a alguma classe de tarefas T e com a medida de desempenho P, se seu desempenho nas tarefas em T, medido por P, melhorar com a experiência E.

Aqui, a Tarefa (T) é avaliar a estética visual de uma tela capturada de um aplicativo Android, Experiência (E) é um *corpus* de imagens de capturas de tela de aplicativos Android com estética visual que varia de feio a bonito em escala numérica [0..1] e seu desempenho (P) é medido pelo tamanho do erro quadrático obtido na predição do valor estético de uma imagem em relação ao valor verdade obtido na pesquisa com os humanos.

A avaliação é feita de interfaces de usuário de aplicativos Android, com base em *screenshots* dessas interfaces. Com exceção de imagens que possuam conteúdo avaliado como potencialmente ofensivo ou inapropriado a algum tipo de público, todas as interfaces são consideradas.

Propõe-se formular a tarefa de avaliação da estética como um problema de regressão com valor real e não como um problema de classificação. Assim, o objetivo é prever pontuações contínuas na classificação da estética de telas de aplicativos para Android que não sejam rótulos de categorias discretas. Considerando a tarefa como um problema de regressão com o objetivo de prever valores contínuos em relação à estética visual, a saída é um valor numérico [0..1] sendo interpretado como o grau de estética visual entre 0=feio e 1=bonito.

4.2 Iterações

O trabalho é desenvolvido durante três iterações, onde as iterações 1 e 2 tratam de uma abordagem de classificação e para a iteração 3 a abordagem é alterada para uma de regressão. Os subcapítulos seguintes descrevem cada iteração:

4.2.1 Iterações 1 e 2 - “Primeiros Resultados”

Este subcapítulo apresenta os resultados obtidos durante nossa primeira e segunda iterações de treinos e testes, nas quais fora utilizada uma abordagem de classificação com a rede neural.

Conjunto de dados

Nesse estudo é usado um conjunto de dados apresentando *screenshots* de interfaces de usuário Android, junto com um label indicando o grau de estética visual da interface.

Os screenshots de interfaces foram retirados de diferentes fontes:

- Imagens de *screenshots* de interfaces de apps Android do Google Play, do conjunto de dados RICO[11].
- Imagens de *screenshots* capturadas de interfaces de apps Android de aplicativos no MIT App Inventor Gallery[12]. Os *screenshots* foram obtidos de forma semiautomática com um Samsung Galaxy S7 e um Sony Xperia XZ.
- Imagens de *screenshots* de interfaces de apps Android de aplicativos criados pelo Grupo de Qualidade de Software (GQS) da Universidade Federal de Santa Catarina (UFSC). Os *screenshots* foram obtidos de forma semiautomática com um Samsung Galaxy S7.

[11] Disponível em: <http://interactionmining.org/rico>
[12] Disponível em: <http://appinventor.mit.edu/explore/>

RICO (bonita)

MIT (mais ou menos)

GQS (bonita)

Figura 32: Interfaces do nosso conjunto de dados (elaborado pelo autor)

Inicialmente, em uma primeira iteração, foi construído um conjunto de dados com 1500 imagens que foi aumentado para 4500 imagens numa segunda iteração, conforme apresentado na Tabela 9. Em todas as iterações, os *screenshots* foram selecionados aleatoriamente, excluindo aqueles que apresentavam conteúdo designado inapropriado e também para evitar ao máximo a inclusão de *screenshots* repetidas.

Tabela 9: Conjuntos de dados (elaborado pelo autor)

Fonte	Iteração 1	Iteração 2
App Inventor (GQS/UFSC)	12	12
App Inventor (*MIT App Inventor Gallery*)[13]	328	328
Conjunto de dados RICO[14]	1160	4160
TOTAL	**1500**	**4500**

Labeling das imagens

Todos os *screenshots* de interfaces de usuário foram categorizados manualmente em relação ao seu grau de estética visual usando uma escala ordinal de 3 pontos (bonita, mais ou menos, feia). Cada screenshot foi avaliado por 3 voluntários do Departamento de Informática e Estatística (INE)/UFSC e do Hiperlab/UFSC. Esse processo foi realizado via

[13] Disponível em: <http://appinventor.mit.edu/explore/>
[14] Disponível em: <http://interactionmining.org/rico>

formulários online (Figura 33). Cada formulário continha 150 *screenshots* e para cada *screenshot* eram disponibilizadas as 3 alternativas de grau de estética visual.

Após coletadas todas as respostas de todos os voluntários, foram analisados os resultados da avaliação manualmente, levando em consideração uma variação de concordância de avaliadores em relação a estética visual (GRESSE VON WANGENHEIM et al., 2018b). Foram excluídas interfaces que não apresentaram uma concordância mínima dos avaliadores. Somente consideramos interfaces nas quais no mínimo 2 avaliadores concordaram com o mesmo grau de estética visual.

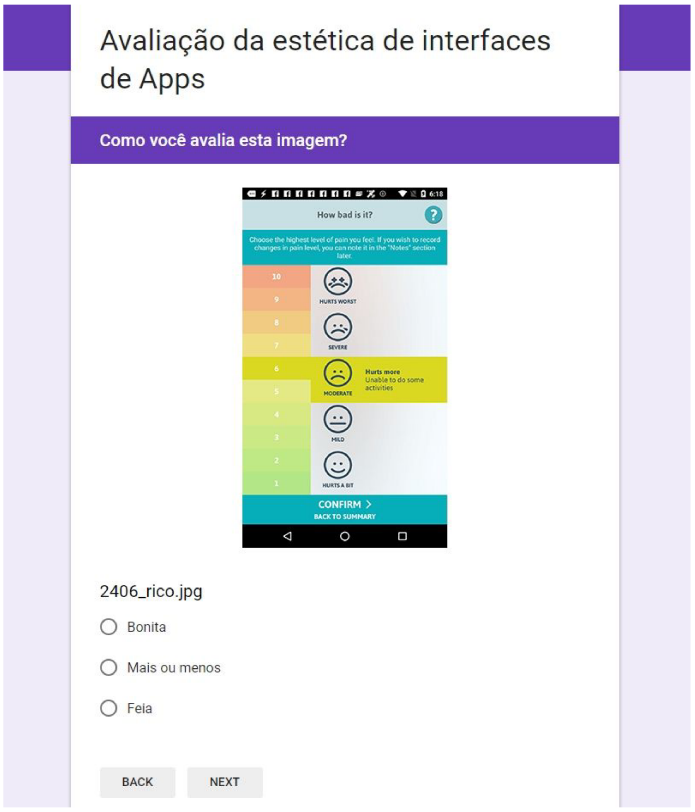

Figura 33: Formulário online (elaborado pelo autor)

Desta forma, a Tabela 10 apresenta a composição final do conjunto de dados com labels em cada iteração.

Tabela 10: Composição (por *label*) dos conjuntos de dados (elaborado pelo autor)

Label	Iteração 1	Iteração 2
Bonita	427	1126
Mais ou menos	218	1105
Feia	570	1592
Indecidida	285	646
Total de *screenshots* utilizados	**1215**	**3854**

Modelo de Aprendizagem

Foi utilizada a biblioteca fast.ai V1, que permite utilizar o *framework* PyTorch 1.0 evitando detalhes de mais baixo nível, permitindo uma prototipação rápida e tornando a construção e treinamento da rede neural mais eficiente (HOWARD, 2018). Fast.ai também disponibiliza estratégias avançadas de otimização de hiperparâmetros (HYPOs) de forma fácil de usar. A lógica por trás dessa decisão de design é baseada no fato deste trabalho ser uma pesquisa em design de interfaces usando técnicas de *deep learning*, em vez de uma pesquisa em *deep learning*.

Os passos realizados durante o modelo de aprendizagem foram:

Seleção do modelo. Os modelos ResNet (*Residual Network*) receberam grande destaque após vencerem competições a respeito de análise de imagens (detecção, categorização, segmentação de objetos, localização, ...) (HE, 2016). ResNets empregam conexões de identidade que atuam como atalhos (Figura 34), ignorando várias camadas ao mesmo tempo, fornecendo dois caminhos de aprendizado paralelo em várias seções da rede, assim evitando a perda de gradiente que normalmente ocorre em redes muito profundas (RUSSAKOVSKY et al., 2015). Modelos ResNets foram escolhidos principalmente por permitirem o uso de HYPOs especialmente desenvolvidos para este modelo de CNN, permitindo um treinamento ainda mais rápido (SMITH, 2018b), (SMITH, TOPIN, 2018).

Adotando uma abordagem de transferência de aprendizado, primeiro foi treinada uma rede básica em um conjunto de dados básico para uma tarefa básica. Quando recebida uma tarefa mais específica, inicializamos as camadas de uma nova rede com os pesos desta rede básica pré-treinada e então ajustamos toda a rede ao conjunto de dados alvo. Esta transferibilidade do conhecimento adquirido aumentará à medida que a distância entre a tarefa alvo e a tarefa base diminuir (YOSINSKI et al., 2014). Com o objetivo de redefinir modelos bases cujas tarefas compartilham uma mesma base com a nossa tarefa de avaliação de estética, o modelo foi pré-treinado com o ImageNet (RUSSAKOVSKY et al., 2014), por ser um dos maiores conjuntos de dados de uso geral disponíveis ao público para o treinamento da classificação de imagens como nossa rede base. As redes utilizadas para testes são apresentadas na Tabela 11.

Tabela 11: ResNet-18, ResNet-34, ResNet-50 (adaptado a partir de HE (2016))

Nome da camada	Tamanho da saída	18-camadas	34-camadas	50-camadas
conv 1	112x112	7x7, 64, *stride* 2		
conv2_x	56x56	3x3, *max pool*, *stride* 2		
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28x28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$
conv4_x	14x14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$
conv5_x	7x7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
average pool	1x1			
fc	1000			
softmax	1000			

As camadas conv2_1, conv3_1, conv4_1 e conv5_1 realizam *downsampling* com passo (*stride)* 2. A Figura 34 ilustra todas as camadas da rede ResNet-34, para melhor visualização da composição destas redes.

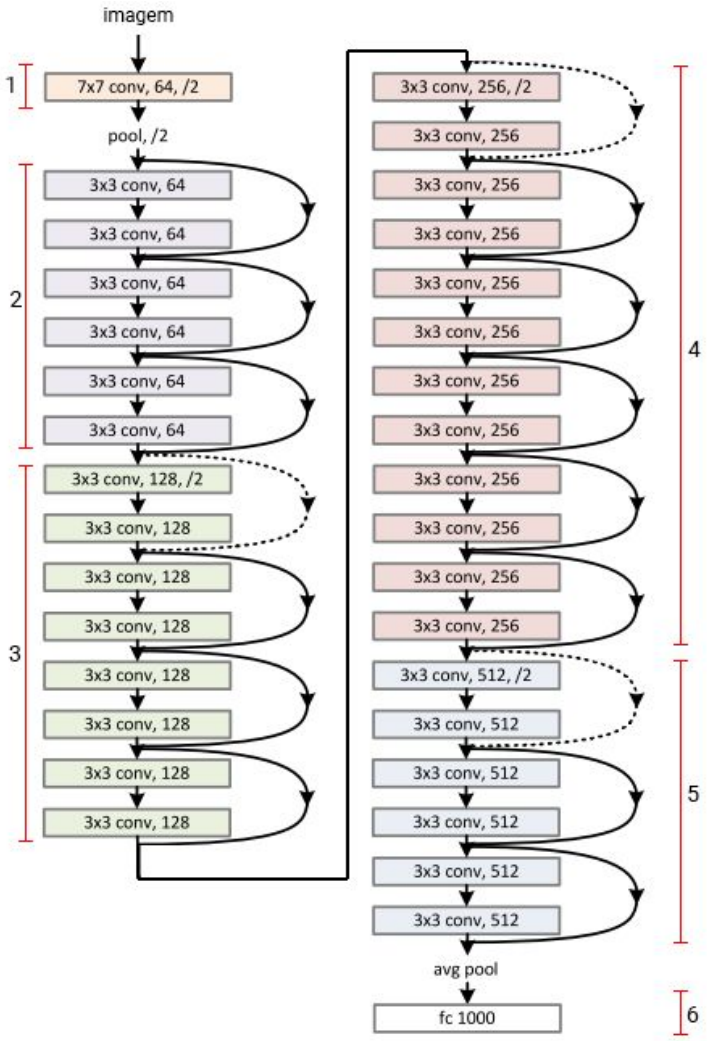

Figura 34: ResNet-34 (HE, 2016)

A camada de entrada da rede (1) é uma camada de convolução com 64 filtros 7x7 e passo 2 para realizar o primeiro *downsampling*. O próximo grupo (2), com 6 camadas, realiza novas operações de convolução e transmite os dados para a segunda operação de *downsampling* na primeira camada do terceiro grupo (3). Após mais 7 camadas de convolução, uma terceira etapa de *downsampling* na primeira camada do quarto grupo (4), seguida de mais 11 camadas de convolução antes da última operação de *downsampling* na primeira camada do quinto grupo (5) de camadas convolucionais. Os dados finais passam por uma operação de *global average pooling* e são então transmitidos para a camada de saída (6); um perceptron totalmente conectado (1000 conexões) com *softmax*, que então gera o resultado numérico final.

Após cada operação de *downsampling* o número de filtros é duplicado para preservar a complexidade temporal por camada.

Seleção e otimização de HYPOs. Escolhemos uma estratégia de otimização para escolha automática de HYPOs chamada de *fit1cycle* (SMITH, 2018b), (SMITH, TOPIN, 2018). Esta estratégia foi especificamente desenvolvida para ResNets e trabalha com taxa de aprendizagem e momentum adaptativos, que variam dinamicamente, seguindo uma curva onde a taxa primeiro é aumentada e então diminuída, enquanto o momentum segue um comportamento oposto (SMITH, 2018b), (SMITH, TOPIN, 2018). As arquiteturas das redes foram adaptadas ao nosso conjunto de dados de treino: a camada de entrada foi adaptada à resolução das imagens em nosso conjunto de dados, representado por um vetor de cada imagem da tela de um aplicativo Android e seu label, representando a pontuação média das classificações feitas pelos usuários em cada tela. As camadas de saída originais também foram substituídas, que nas redes pré-treinadas pelo ImageNet representam uma variável categórica com 1000 valores (contendo 1000 neurônios para o conjunto de dados ImageNet) por uma camada de regressão com um único neurônio de saída.

Transferência de aprendizado. Como o conjunto de dados é relativamente pequeno, o que pode trazer riscos como *overfitting* para o modelo, foi explorada a transferência de aprendizado a partir de um ponto de partida com inicialização de uma rede já pré-treinada. Essa estratégia permite o treino de uma CNN com dados limitados efetivamente.

Treinamento. A fase de *fine-tuning* foi realizada usando a mesma estratégia da fase de transferência de aprendizado, apenas descongelando e permitindo a adaptação de todos os pesos da rede neural. Ajustes foram realizados nas redes utilizando a estratégia *fit1cycle* mencionada anteriormente, que permite ajustes automáticos de HYPOs.

4.2.1.1 Resultados

Nessa seção são apresentados os primeiros resultados obtidos nos testes durante 2 iterações com diferentes tamanhos do conjunto de dados .

Iteração 1

O conjunto de dados foi dividido em dois conjuntos: 80% (972 imagens) para treinamento (*Train*) e 20% (243 imagens) para validação (*Validation*) de maneira aleatória.

Os primeiros resultados obtidos com a rede ResNet-34 são observados na Figura 35.

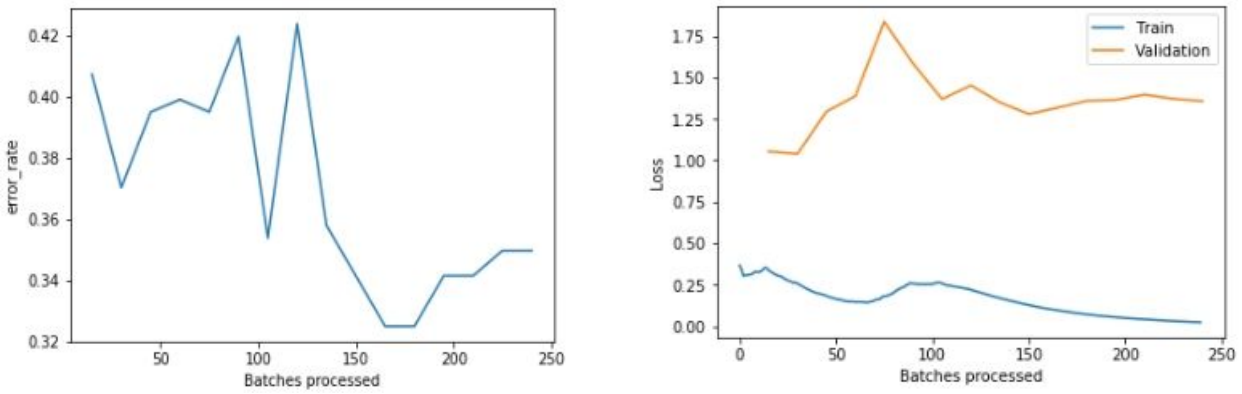

Figura 35: Iteração 1, gráficos de erro e perda ResNet-34 (elaborado pelo autor)

A alta flutuação no gráfico de erros e a divergência no gráfico de perda sugere que a rede não foi capaz de generalizar o conhecimento do treinamento. A queda no índice de perda para o conjunto de treino e aumento do índice no conjunto de validação suporta este argumento, mostrando que a rede, na verdade, “memorizou” os dados de treinamento.

A seguir, novos testes foram realizados com uma rede menos profunda: ResNet-18. O conjunto de dados foi novamente separado de maneira aleatória, mas mantendo as mesmas quantidades de imagens para ambos os conjuntos. A Figura 36 apresenta os resultados obtidos com a ResNet-18, enquanto a Figura 37 apresenta exemplos de mapas de calor obtidos pela rede.

epoch	train_loss	valid_loss	error_rate	time
0	1.279274	1.068755	0.584362	02:41
1	1.198799	0.943850	0.452675	00:18
2	1.079978	0.891603	0.395062	00:17
3	0.961728	0.904056	0.366255	00:17
4	0.847291	0.928980	0.362140	00:17
5	0.730350	0.958808	0.366255	00:17
6	0.645444	0.994560	0.366255	00:17
7	0.567994	1.033490	0.341564	00:16
8	0.494451	1.111364	0.345679	00:17
9	0.431651	1.169583	0.358025	00:17
10	0.373795	1.127918	0.353909	00:17
11	0.325569	1.210161	0.353909	00:17
12	0.279115	1.241418	0.366255	00:16
13	0.237351	1.312783	0.362140	00:16
14	0.210375	1.347055	0.349794	00:16
15	0.182545	1.376725	0.349794	00:16
16	0.157471	1.276173	0.362140	00:16
17	0.140019	1.344885	0.349794	00:16
18	0.122441	1.376403	0.325103	00:16
19	0.104247	1.367476	0.333333	00:16

Figura 36: Iteração 1, treino ResNet-18 (elaborado pelo autor)

Percebe-se que o mesmo comportamento se repetiu com a arquitetura ResNet-18; a taxa de perda para o conjunto de treino (*train_loss*) diminui a cada etapa, mas a taxa de perda para o conjunto de validação (*valid_loss*) aumenta. Novamente a rede “memorizou” o treinamento e não generalizou o conhecimento.

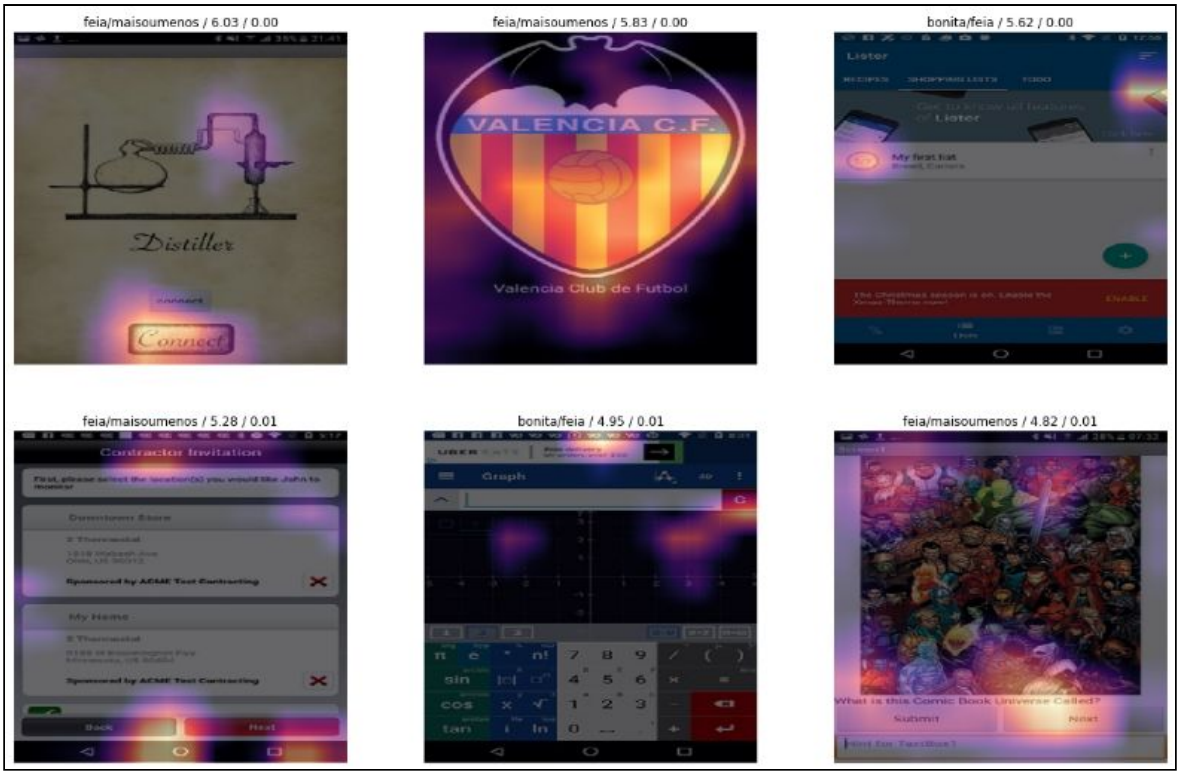

Figura 37: Iteração 1, mapas de calor ResNet-18 (elaborado pelo autor)

Os mapas de calor não oferecem informações relevantes, a rede parece não saber para o que olhar. Como passo seguinte decidiu-se encerrar a iteração 1 e aumentar o conjunto de dados em preparação para uma segunda iteração.

Iteração 2

Após atualização do conjunto de dados conforme descrito na seção 4.2 deu-se início à nova fase de testes. O conjunto de dados foi novamente dividido em 80% (3084 imagens) para o conjunto de treinamento (Train) e 20% (770 imagens) para o conjunto de validação (Validation) de maneira aleatória.

A Figura 38 ilustra os gráficos de erro e perda para a ResNet-34 após a conclusão do treinamento.

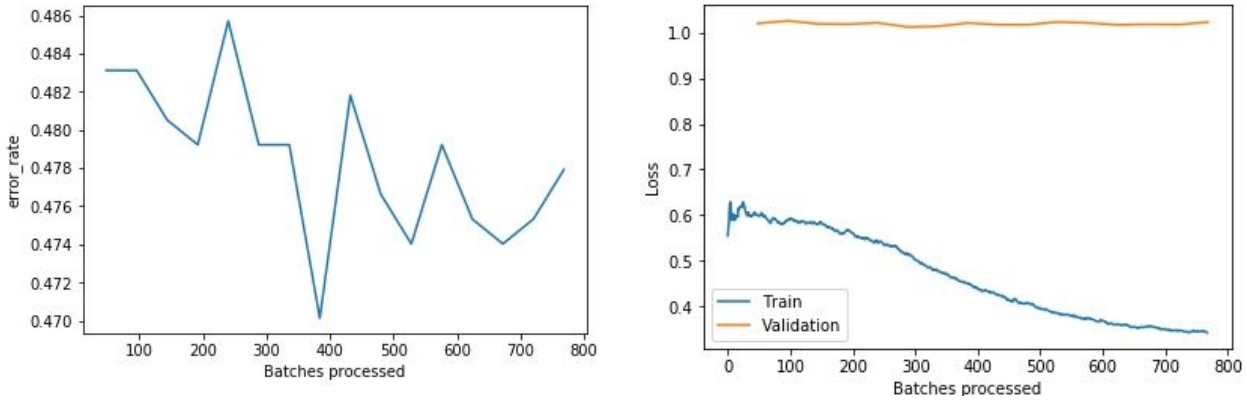

Figura 38: Iteração 2, gráficos de erro e perda ResNet-34 (elaborado pelo autor)

O mesmo comportamento dos testes anteriores é observado aqui; a taxa de erro flutua consideravelmente enquanto a taxa de perdas para o conjunto de treinamento cai enquanto a taxa de erros para o conjunto de validação não demonstra declínio, novamente indicando “memorização” do conjunto de dados pela rede ao invés da generalização do aprendizado.

A Figura 39 relata os resultados obtidos com a rede ResNet-18 nesta segunda iteração, porém, novamente nenhuma melhora pode ser observada.

epoch	train_loss	valid_loss	error_rate	time
0	1.273015	1.026742	0.527273	00:49
1	1.147959	0.987941	0.485714	00:49
2	1.039501	1.013088	0.477922	00:48
3	0.940943	1.021937	0.490909	00:49
4	0.860308	1.022211	0.493506	00:48
5	0.784060	1.046155	0.471429	00:48
6	0.721257	1.057554	0.500000	00:48
7	0.660072	1.068633	0.471429	00:48
8	0.583460	1.100964	0.489610	00:49
9	0.506947	1.122296	0.488312	00:49
10	0.428909	1.172363	0.489610	00:49
11	0.374269	1.188859	0.493506	00:49
12	0.324751	1.193223	0.474026	00:49
13	0.284387	1.212692	0.470130	00:49
14	0.262657	1.223246	0.471429	00:49
15	0.255280	1.224611	0.474026	00:49

Figura 39: Iteração 2, treino ResNet-18 (elaborado pelo autor)

Como os resultados não foram satisfatórios nestas duas iterações, foi decidido adotar uma abordagem diferente para o problema. O modelo de classificação foi adaptado para agora trabalhar com regressão e assim deu-se início à iteração 3.

4.2.2 Iteração 3

Neste subcapítulo são apresentados os resultados da terceira, e última, iteração de treinamento e testes realizados neste trabalho, na qual passa-se a utilizar uma abordagem de regressão com a rede neural. Esta decisão veio da ideia de que predições incorretas feitas pela rede deveriam possuir “pesos” diferentes. Por exemplo, ao avaliar uma imagem cujo valor verdade é 0,87, uma predição de 0,4 deveria ser percebida como pior do que uma predição de 0,6 mesmo que ambas estejam erradas. O modelo de regressão possibilita este tipo de comportamento (DOU et al., 2019).

Conjunto de Dados

Neste estudo é usado um conjunto de dados apresentando *screenshots* de interfaces de usuário Android, junto com um label indicando o grau de estética visual da interface.

Coleta dos dados. Nenhum novo conjunto de *screenshots* foi coletado para esta terceira iteração em relação à anterior.

Limpeza dos dados/pré-processamento. Eliminou-se todas as duplicatas, e selecionou-se apenas imagens consideradas aceitáveis em relação a questões éticas. Para a iteração 3 também foram consideradas apenas imagens sem propagandas e foi feita a remoção da barra de status do Android localizada na parte superior das capturas de tela, para evitar variações indesejadas não diretamente relacionadas às próprias interfaces do usuário. Também foi feita uma redução de escala para 239x408 pixels para padronizar todas as imagens.

Tabela 12: Tabela atualizada do conjunto de dados (elaborado pelo autor)

Fonte	Iteração 1	Iteração 2	Iteração 3
App Inventor (GQS/UFSC)	12	12	11
App Inventor (*MIT App Inventor Gallery*)[15]	328	328	252
Conjunto de dados RICO[16]	1160	4160	2876
TOTAL	**1500**	**4500**	**3139**

Labeling das imagens

Para a iteração 3 foi realizada uma transformação dos labels para uma escala racional de 0 a 1. Para isto os votos recebidos por cada imagem foram relacionados a um valor numérico, com:

- Bonita = 1,0
- Mais ou menos = 0,5
- Feia = 0,0

Foi calculada a média dos valores e atribuídas às imagens como seu novo label. Esta é uma preparação para se trabalhar com um modelo de regressão, pois com este modelo a rede é capaz de medir o "tamanho de um erro". Por exemplo, uma predição de um valor 0,7 com alvo 0,4 deve ser pior que uma inferência 0,5 com alvo 0,4. Caso seja empregada a perda de entropia cruzada tradicional, que formula a tarefa como um apenas um problema de classificação, as predições erradas geram perdas iguais independentes do quão ruim foi o erro.

[15] Disponível em: <http://appinventor.mit.edu/explore/>
[16] Disponível em: <http://interactionmining.org/rico>

Devido à eliminação das imagens que receberam três avaliações distintas pelos humanos, observa-se uma redução na quantidade de imagens com pontuação média final de 0,5. Contudo, a distribuição dos dados remanescentes permanece balanceada, o que deverá ajudar a rede neural a adquirir um conhecimento balanceado do conjunto como um todo. Assim, a Figura 40 mostra a distribuição final de todas as 3.139 imagens de acordo com suas pontuações médias.

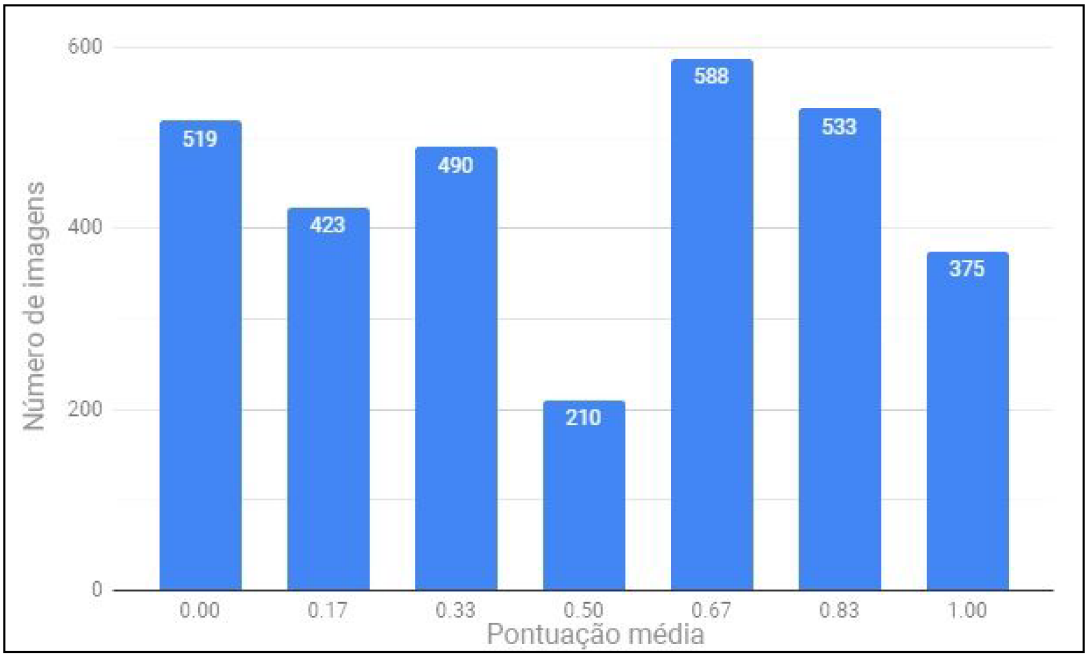

Figura 40: Distribuição das imagens por pontuação média (elaborado pelo autor)

Divisão dos dados. Os dados foram divididos de maneira aleatória em:

- Conjunto de treino: conjunto com 2.509 imagens (80%) usadas para treino, isto é, para ajustar os parâmetros do classificador.
- Conjunto de teste: conjunto com 630 imagens (20%), com 90 imagens de cada grupo da média de valores, usados para ajustar os parâmetros de um classificador, como por exemplo, para escolher o número de unidades ocultas em uma rede neural. Foi decidido manter o conjunto de teste balanceado para melhor análise estatística dos erros.

A divisão foi feita com um *script* em Python, apresentado no Apêndice A. Como as imagens no conjunto já se encontram distribuídas aleatoriamente, o *script* apenas precisa selecionar uma sequência qualquer de imagens para montar o conjunto de teste. Porém, foi também mantido um controle das imagens selecionadas para se manter uma distribuição balanceada em relação à média do valor de cada imagem. As Figuras 41 e 42

apresentam a distribuição do conjunto de treino e de testes, respectivamente. Com esta distribuição balanceada torna-se mais simples realizar estudos estatísticos sobre os erros e acertos das predições do modelo.

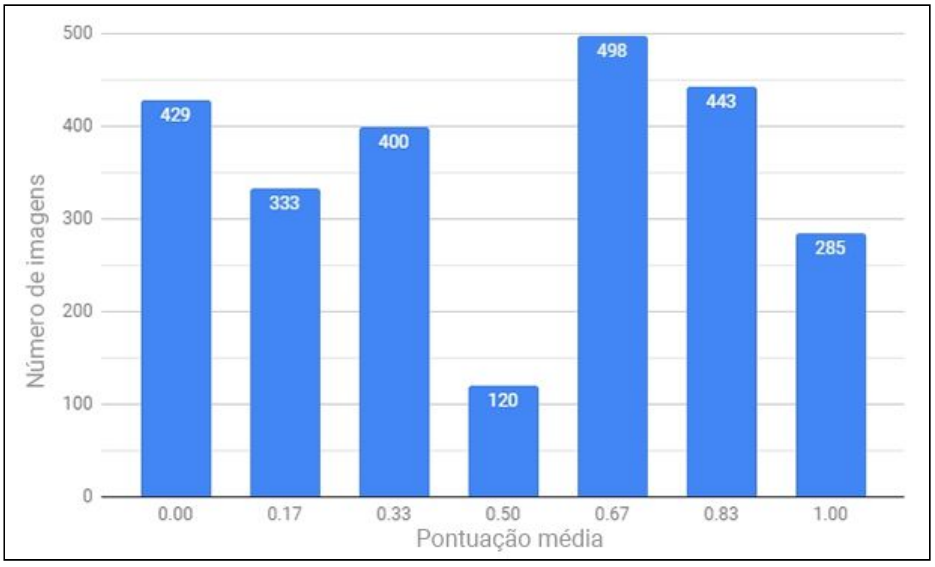

Figura 41: Distribuição das imagens por pontuação média - conjunto de treino (elaborado pelo autor)

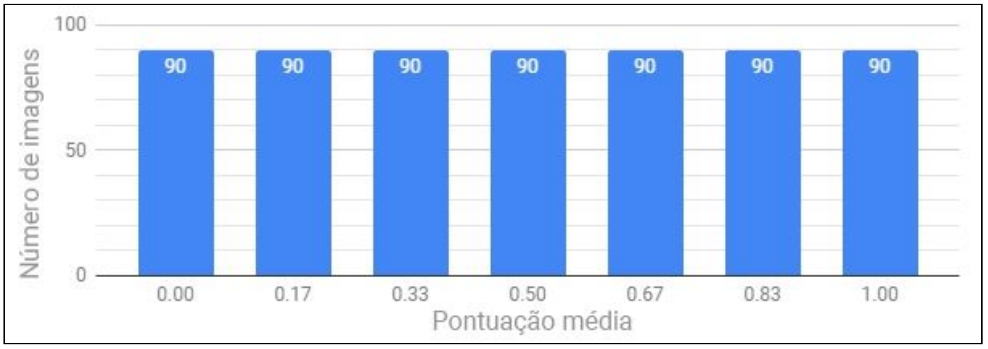

Figura 42: Distribuição das imagens por pontuação média - conjunto de testes (elaborado pelo autor)

O conjunto de dados completo está disponível publicamente em: http://www.lapix.ufsc.br/mobile-interfaces-dataset/

Modelo de Aprendizagem

Não houve alterações nas bibliotecas e *frameworks* utilizados nesta iteração em relação à iteração anterior.

Os passos realizados durante o modelo de aprendizagem foram:

Seleção do modelo. Os modelos utilizados são os mesmos das iterações anteriores, porém, com a adição de um modelo mais poderoso (ResNet-101) para melhores comparações dos resultados obtidos. A Tabela 12 traz a tabela atualizada com todos os modelos agora utilizados.

Tabela 13: ResNets-18, 34, 50 e 101 (adaptado a partir de HE (2016))

Nome da camada	Tamanho da saída	18-camadas	34-camadas	50-camadas	101-camadas
conv 1	112x112	7x7, 64, *stride* 2			
conv2_x	56x56	3x3, *max pool*, *stride* 2			
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28x28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14x14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7x7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
average pool	1x1				
fc	1000				
softmax	1000				

Seleção e otimização de HYPOs. As estratégias de otimização permanecem as mesmas das iterações anteriores, utilizando do ajuste automático de HYPOs com a estratégia *fit1cycle* (SMITH, 2018b), (SMITH, TOPIN, 2018). Contudo, todas as redes foram treinadas de duas maneiras, para comparações de resultados: empregando esta seleção automática de HYPOs e também com HYPOs fixos. Além disso, a perda de entropia cruzada da classificação é alterada para perda de regressão, modificando a perda de *softmax* para uma perda de erro quadrático médio. O resultado, dada uma imagem de entrada de uma *screenshot* de uma interface de usuário de um aplicativo Android, é, portanto, uma pontuação estética prevista como um número real que varia entre 0 e 1.

Transferência de aprendizado. A estratégia de transferência de aprendizado também permanece a mesma em relação às iterações anteriores.

Treinamento. A fase de *fine-tuning* foi realizada usando a mesma estratégia da fase de transferência de aprendizado, apenas descongelando e permitindo a adaptação de todos os pesos da rede neural. Ajustes foram realizados nas redes de duas maneiras: utilizando a estratégia *fit1cycle* mencionada anteriormente, que permite ajustes automáticos de HYPOs, e também uma estratégia sem ajustes automáticos. A Tabela 13 resume as redes treinadas com ambas estratégias e os resultados observados usando o conjunto de teste para a avaliação.

Tabela 14: Resumo do treinamento do modelo, estratégias e resultados observados (elaborado pelo autor)

Modelo	**Treino**	**Fase de ajustes (*fine-tuning*)**	**Melhor erro quadrático médio**
ResNet18	HYPOs fixos	Descongelada; Taxa de aprendizado máxima limitada para aprendizagem ótima;	0.078778
	Ajuste automático de HYPOs	Descongelada; Taxa de aprendizado máxima limitada para aprendizagem ótima;	**0.075770**
ResNet34	HYPOs fixos	Descongelada; Taxa de aprendizado máxima limitada para aprendizagem ótima;	0.078778
	Ajuste automático de HYPOs	Descongelada; Taxa de aprendizado máxima limitada para aprendizagem ótima;	**0.065101**
ResNet50	HYPOs fixos	Descongelada; Taxa de aprendizado máxima limitada para aprendizagem ótima;	0.070096
	Ajuste automático de HYPOs	Descongelada; Taxa de aprendizado máxima limitada para aprendizagem ótima;	**0.051369**
ResNet101	HYPOs fixos	Descongelada; Taxa de aprendizado máxima limitada para aprendizagem ótima;	0.082701
	Ajuste automático de HYPOs	Descongelada; Taxa de aprendizado máxima limitada para aprendizagem ótima;	**0.052089**

As Figuras 43 e 44 trazem os resultados do melhor treino obtido com a ResNet50.

epoch	train_loss	valid_loss	mean_squared_error	time
0	3.717225	1.653023	1.653023	10:41
1	2.259219	0.662116	0.662116	01:18
2	0.858105	0.225350	0.225350	01:18
3	0.321166	0.112725	0.112725	01:17
4	0.173703	0.099027	0.099027	01:19
5	0.122845	0.087572	0.087572	01:19
6	0.103133	0.099738	0.099738	01:19
7	0.094555	0.083378	0.083378	01:19
8	0.089664	0.084625	0.084625	01:19
9	0.081430	0.083917	0.083917	01:19

Figura 43: Treino ResNet50 com pesos congelados (elaborado pelo autor).

epoch	train_loss	valid_loss	mean_squared_error	time
0	0.064045	0.051369	0.051369	03:33
1	0.061142	0.051531	0.051531	03:32
2	0.058698	0.051369	0.051369	03:32

Figura 44: Treino ResNet50 com pesos descongelados (elaborado pelo autor).

O modelo desenvolvido encontra-se disponível em: https://codigos.ufsc.br/aldo.vw/appsthetics

5. AVALIAÇÃO DO MODELO

O objetivo deste trabalho é prever classificações sobre a estética visual da interface de usuário de aplicativos Android que correspondam às classificações dos usuários reais. Normalmente, para esse tipo de avaliação, são utilizados estudos de correlação e regressão. Portanto, visando comparar nossos resultados com os de outros trabalhos correlatos, também utilizou-se do teste de correlação de Pearson, seguindo a avaliação realizada por Dou et al. (2019).

No entanto, como a correlação estuda a relação entre uma variável e outra, e não as diferenças, não é recomendado como método para avaliar a comparabilidade entre os métodos (GIAVARINA, 2015). Correlação quantifica o grau em que duas variáveis estão relacionadas, não a concordância entre elas. Portanto, o coeficiente de correlação e a técnica de regressão às vezes são inadequados e podem ser enganosos ao avaliar a concordância, porque avaliam apenas a associação linear de dois conjuntos de observações (GIAVARINA, 2015). Por outro lado, a análise de Bland-Altman é baseada na quantificação da concordância entre duas medidas quantitativas, estudando a diferença média e construindo limites de concordância. Bland e Altman introduziram o gráfico de Bland-Altman (B&A) para descrever a concordância entre duas medidas quantitativas (BLAND; ALTMAN, 1986). Assim, a análise de plotagem de B&A permite avaliar um viés entre as diferenças médias e estimar um intervalo de concordância, dentro do qual caem 95% das diferenças do segundo método, comparado ao primeiro. O B&A recomenda que 95% dos pontos de dados fiquem dentro de ± 2s da diferença média. Os dados podem ser analisados tanto como gráfico de diferenças de unidade quanto como gráfico de diferenças de porcentagem. Portanto, também realizou-se uma análise de exatidão e precisão em termos da análise de Bland-Altman (BLAND; ALTMAN, 1986).

Desta forma, cada uma dessas análises são apresentadas nas próximas seções.

5.1 Análise de Correlação

O modelo Appsthetics, treinado para a avaliação estética da interface de usuários de aplicativos Android, foi avaliado com o conjunto de dados de teste com 630 imagens conforme apresentado na Seção 4.2.2. Mediu-se a força de associação linear entre os resultados da rede e os valores verdade baseados nas avaliações humanas por meio do coeficiente de correlação de Pearson (*r)* (PECK; OLSEN; SHORT, 2019) que é calculado

como a razão de covariância entre as variáveis e o produto de seus desvios padrão. O valor numérico de r varia de -1,0 a +1,0. Quanto mais próximo de 1,0 maior a correlação positiva (ambos valores tendem a aumentarem juntos), enquanto mais próximo de -1,0 maior a correlação negativa (quando um valor aumenta o outro tende a diminuir). Um valor muito próximo ou igual a 0,0 indica que não há uma associação entre os dois valores.

Os resultados da análise de Pearson são apresentados na Figura 45.

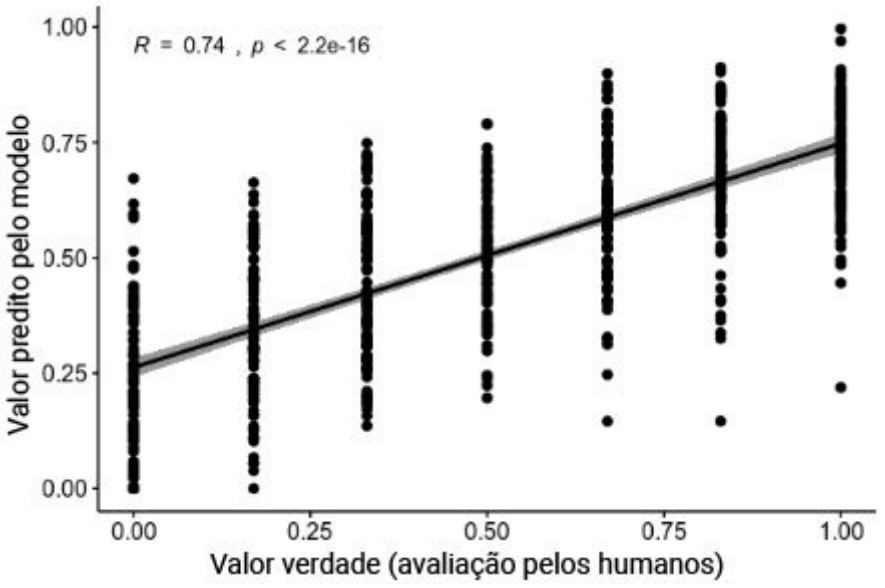

Figura 45: Gráfico da análise de Pearson (elaborado pelo autor)

Quantifying webpage aesthetics, Dou et al. (2019), relata uma correlação de Pearson de (r=0,85, p < 0,001). Em comparação, nossa abordagem com *deep learning* obteve um fator de correlação de Pearson r=0,74 e p<0,00001. A Tabela 14 relata os valores Pearson obtidos em nossa análise e os publicados nos trabalhos correlatos.

Tabela 15: Comparação dos valores de correlação de Pearson (elaborado pelo autor)

Trabalho	**Valor de correlação Pearson**
(KHANI et al., 2016)	Não informado
(DOU et al., 2019)	r = 0,85, p < 0,001
Appsthetics	r = 0,74, p < 0,00001

Apesar de obter um valor de correlação de Pearson inferior ao observado em Dou et al. (2019), r = 0,74 ainda demonstra uma boa correlação entre as predições e os valores verdades, e o nosso valor p = 0,00001 indica que o valor de correlação é significante. Como Khani et al. (2016) não informou um valor de correlação Pearson, não podemos realizar uma comparação entre o nosso resultado e o seu trabalho.

5.2 Análise de Bland-Altman

O modelo Appsthetics, treinado para a avaliação estética da interface de usuários de aplicativos Android, foi avaliado com o conjunto de dados de teste. Analisou-se a diferença média quantificando a concordância entre os resultados do nosso modelo (ResNet50) e os valores verdade baseados nas avaliações humanas por meio da análise de Bland-Altman.

O sistema de plotagem B&A não diz se a concordância é suficiente ou até mesmo adequada para se usar um método ou o outro. Ele simplesmente quantifica o viés e uma faixa de concordância, dentro da qual 95% das diferenças entre uma medição e outra estão incluídas. Deve-se sempre definir *a priori* os limites das diferenças máximas aceitáveis (limites de concordância esperados), com base em critérios relevantes específicos do domínio e, em seguida, obter estatísticas para verificar se esses limites foram ou não excedidos.

Os resultados da análise de Bland-Altman são apresentados na Figura 46.

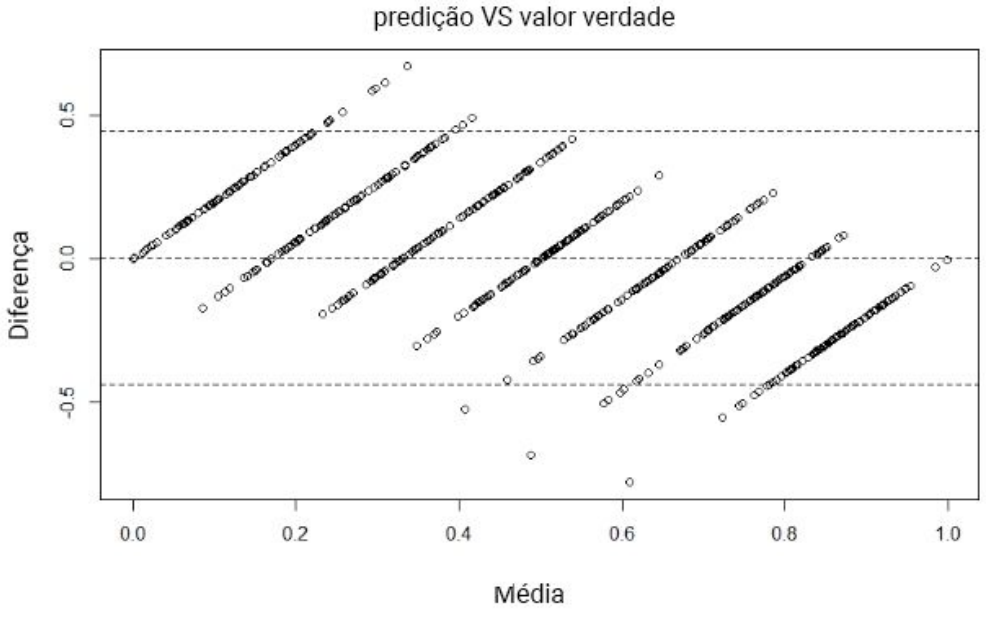

Figura 46: Gráfico da análise de Bland-Altman (elaborado pelo autor)

Observa-se que a análise de Bland-Altman indica uma alta concordância entre o valor verdade e os valores previstos da avaliação da estética visual, com um viés próximo de zero (0,0051).

Como os trabalhos correlatos não apresentaram uma análise por Bland-Altman, não é possível uma comparação.

5.3 Comparação dos resultados da avaliação

Com o objetivo de tentar entender os resultados das análises, examinamos exemplos de avaliação com alta correspondência e baixa correspondência entre a previsão do modelo e as avaliações humanas. A Figura 47 ilustra vários exemplos específicos de telas de aplicativos Android com as pontuações de classificação do usuário (valores verdade) e também as pontuações estéticas previstas pelo modelo Appsthetics.

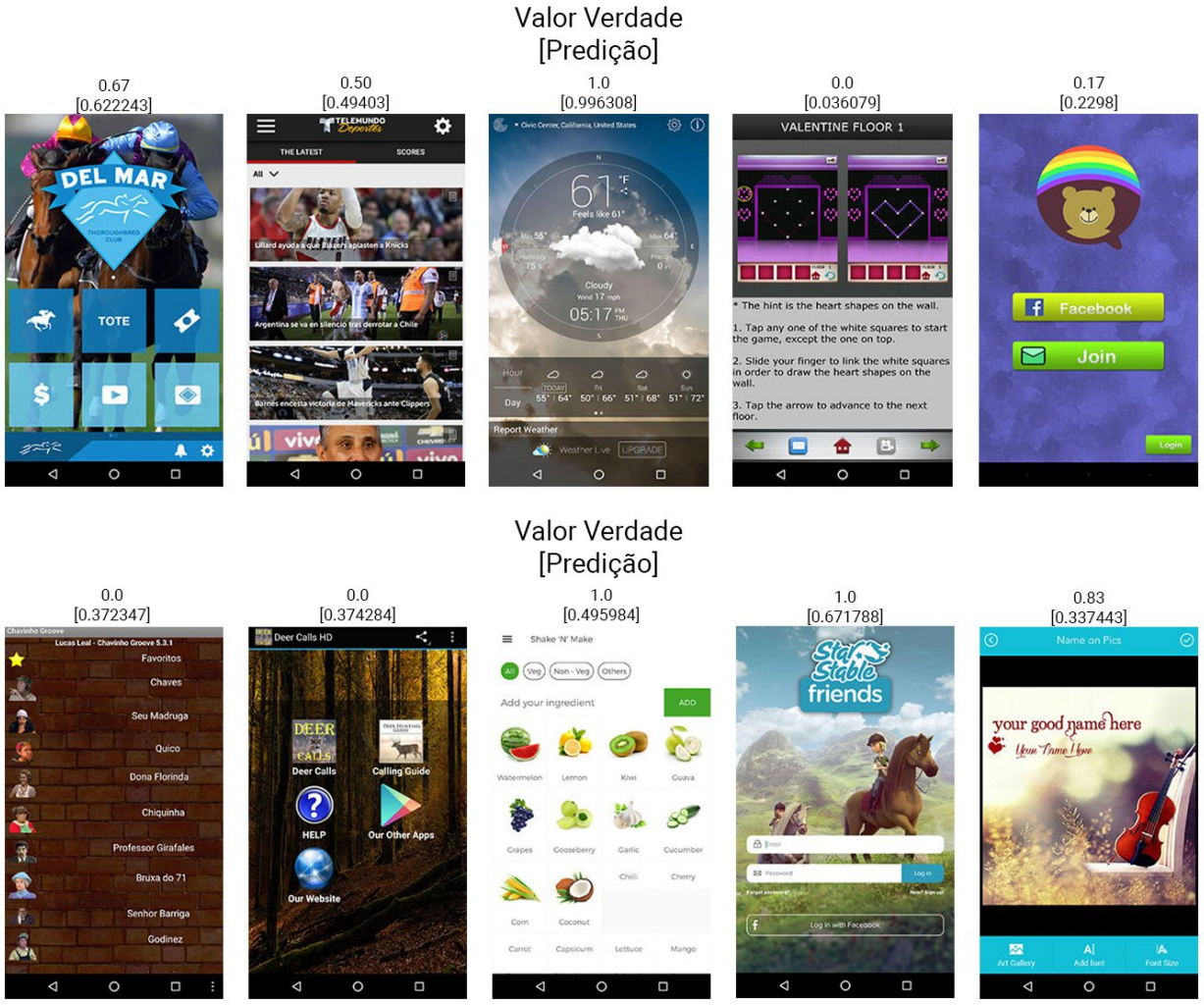

Figura 47: Telas com boas previsões (superior) e más previsões (inferior) (elaborado pelo autor)

Observa-se que o modelo Appsthetics parece atribuir pesos maiores às cores e suas variações durante suas previsões, como visto na linha inferior da Figura 47. Embora os humanos tenham dado avaliações baixas para as duas primeiras imagens, o modelo previu um valor consideravelmente mais alto, provavelmente devido às suas cores. Da mesma forma, ainda na linha inferior, as duas últimas imagens receberam altas avaliações pelos humanos mas baixas pelo modelo. A Figura 48 apoia esta explicação ainda mais, exibindo como o modelo atribui pontuações baixas a imagens com conteúdo predominantemente branco e pontuações consideravelmente maiores a imagens mais escuras ou com poucas variações de tom.

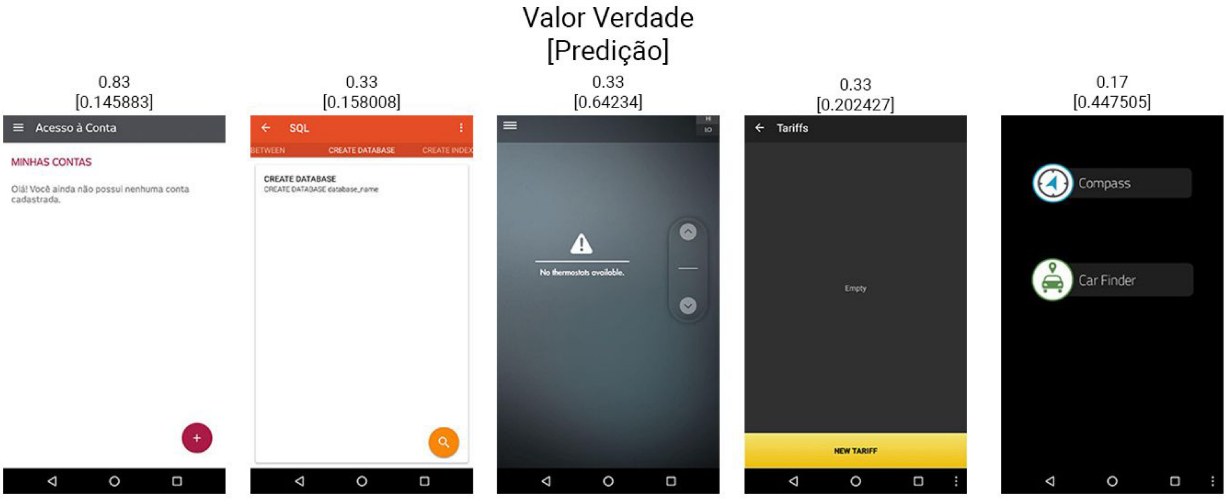

Figura 48: Telas brancas, escuras e pouca variação de cores (elaborado pelo autor)

Vale ressaltar que até mesmo nas avaliações humanas observou-se uma falta de consenso em relação ao nível de estética das interfaces de usuários, e portanto, sugere-se como um trabalho futuro, uma melhor análise destes resultados visando entender exatamente a causa do desvio das predições em relação ao valor verdade obtido nas avaliações humanas.

6. ***DEPLOYMENT***

Visando a utilização do modelo Appsthetics no contexto de uso de computação por meio de desenvolvimento de apps com App Inventor, Esta seção documenta o processo de *deployment* do nosso modelo junto ao sistema CodeMaster (GRESSE VON WANGENHEIM et al., 2018a). CodeMaster é um sistema gratuito disponível online para

avaliar automaticamente projetos desenvolvidos com App Inventor (e Snap!). Ele se concentra na avaliação educacional de atividades de programação complexas e não estruturadas, sem solução correta única, como por exemplo estudantes desenvolvendo seus próprios aplicativos (GRESSE VON WANGENHEIM et al., 2018a). Atualmente o CodeMaster suporta a avaliação de conceitos de algoritmos e programação (ALVES et al., 2019) e a conformidade do design visual com diretrizes (SOLECKI et al., 2019). A análise desses critérios é feita por meio de uma análise estática do código (arquivo .aia) com base em rubricas definidas e estatisticamente validadas.

6.1 Análise de Requisitos da Integração

A Tabela 15 apresenta os requisitos funcionais e a Tabela 16 apresenta os requisitos não funcionais identificados conforme a necessidade de incluir a funcionalidade de avaliação da estética visual da interface de usuário de aplicativos Android ao sistema CodeMaster.

Tabela 16: Requisitos funcionais (elaborado pelo autor)

Requisito	**Descrição**	**Artefato Entrada**	**Artefato Saída**
Avaliar a estética visual da interface de usuário	A ferramenta deve avaliar o grau de estética visual da interface de usuário e apresentar uma nota representando o grau de estética visual.	Arquivo .jpg, com resolução máxima de 1024 pixels de altura e orientação vertical (retrato)	Nota de 0 a 10 representando o grau da estética visual

Tabela 17: Requisitos não funcionais (elaborado pelo autor)

Requisito	**Descrição**
Descartar imagens avaliadas	Por não termos um controle do tipo de informação que por um acaso possa haver na imagem selecionada pelo Aluno, a mesma pode conter material sensível como fotografia de menores de idade ou imagens e ícones sem a devida permissão de uso. Portanto, não devemos armazená-las no sistema.
Linguagem de Programação Java	A inclusão da nova função no sistema CodeMaster deve ser realizada com a linguagem de programação Java 8 para manter homogeneidade com o sistema já existente.
Linguagem de Programação Python	O pequeno servidor que hospedará a rede neural deve ser desenvolvido na linguagem de programação Python 3.7, pois esta é a linguagem nativa da própria rede neural.
Internacionalização	O sistema deve conter a opção de textos em português e inglês.

6.2 Definição do Caso de Uso

Conforme o sistema existente do CodeMaster e a aprimoração apresentada nos requisitos, foram elaborados os casos de uso seguindo Wazlawick (2016).

Caso de Uso 1 - USC01 Avaliação da estética de interface de usuário

Ator: Aluno

Fluxo principal:

1. Aluno acessa o site do CodeMaster
2. Aluno clica no item de menu “Aluno”.
3. Aluno clica em “Escolher arquivo” e escolhe uma ou mais imagens de *screenshots* de telas de usuário salvas em seu computador.
4. Aluno clica em “Avaliar”.
5. O sistema retorna o resultado da avaliação da estética visual da interface de usuário na aba de “Interface de Usuário”, abaixo dos critérios de design.

Fluxo alternativo:

3a. Aluno clica em “Escolher arquivo” e escolhe projeto AppInventor salvo em seu computador.

4a. Aluno clica em “Avaliar” e sistema retorna o resultado da análise do projeto AppInventor.

5a. Aluno clica na aba de “Interface de Usuário”.

6a. Aluno clica em “Escolher arquivo” no campo de “Estética visual” e escolhe uma ou mais imagens de *screenshots* de telas de usuário salvas em seu computador.

7a. Aluno clica em “Avaliar” e sistema retorna o resultado da avaliação da estética visual da interface de usuário na aba de “Interface de Usuário”, abaixo dos critérios de design.

Fluxo de exceção:

FE01. Sistema não consegue analisar a imagem enviada e uma página informando que um “Erro ocorreu” com a opção de retornar à tela inicial é exibida.

FE02. Aluno clica em “Avaliar” após selecionar um (ou mais) arquivos com tamanho excedente ao limite informado e um aviso informando que “Tamanho total dos arquivos selecionados excede o limite disponível” é exibido.

6.3 Modelagem e Implementação

A ferramenta CodeMaster é dividida em 3 grandes componentes. O objetivo da divisão é para permitir que a ferramenta seja escalável a longo prazo e permitir a conexão direta de outras aplicações no futuro. Os componentes são:

- Container Web 1: contém o módulo de apresentação e acesso ao banco de dados.
- Container Web 2: contém o módulo de análise e avaliação de código.
- Navegador de internet: consiste no navegador de internet do usuário, onde a interface gráfica é exibida.

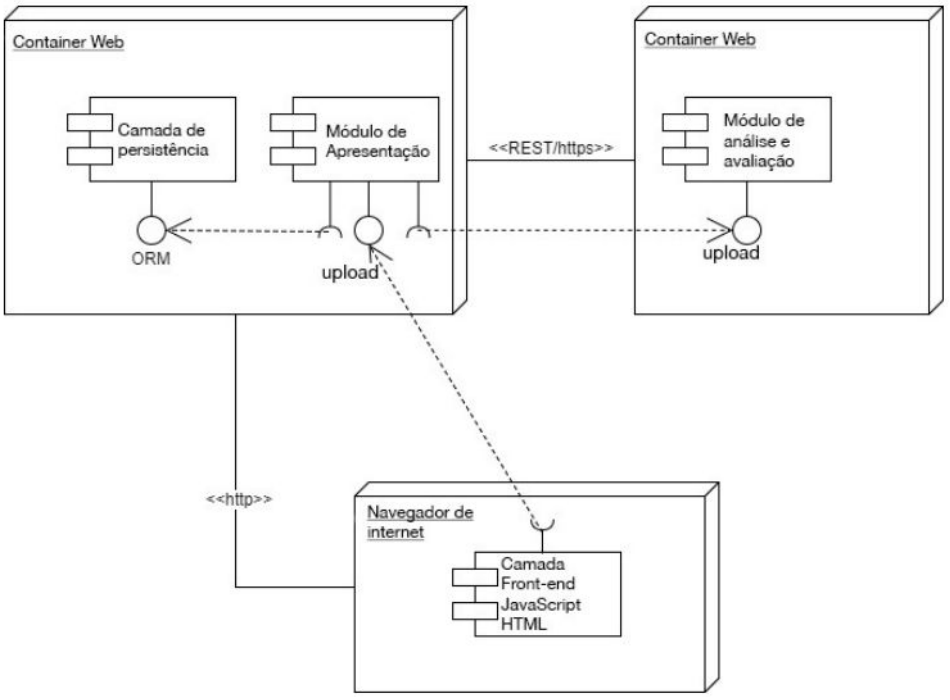

Figura 49: Diagrama de componentes (DEMETRIO, 2017)

O módulo “Apresentação” é responsável pelo controle da interface de usuário, o registro de professores e turmas, a submissão de projetos e a apresentação de resultados para professores e alunos.

O módulo “Front-end” tem como função o desenvolvimento das páginas web apresentadas no navegador de internet do usuário. Esse módulo utiliza JSP, Java Script, HTML5 e CSS3.

O módulo "Análise e avaliação" é responsável em analisar e avaliar o(s) aplicativo(s) conforme os critérios de interesse. O(s) aplicativo(s) e o arquivo de configuração são

fornecidos pelo módulo “Apresentação”. E módulo "Análise e avaliação" envia para o módulo “Apresentação” o(s) feedback(s) instrucional(ais) e a pontuação alcançada em cada critério. Essa comunicação é feita por meio de um serviço web *Representational State Transfer* (REST). REST é uma arquitetura que permite a separação de sistemas web em módulos (RODRIGUEZ, 2008). Para implementar esse serviço o CodeMaster utiliza o *framework* Jersey (2019), pois abstrai os detalhes de baixo nível da implementação de comunicação entre os servidores e simplifica a implementação de um serviço REST. A escolha do *framework* REST foi devido a simplicidade de uso, por ser adequado ao que é proposto no CodeMaster e pela grande quantidade de material disponível (DEMETRIO, 2017).

A inclusão da função de avaliação da estética visual de interfaces de usuário de aplicativos Android implica em alterações nos módulos “Apresentação” e “Front-end”, bem como a introdução de um novo módulo “Appsthetics”. Esse novo módulo conterá um mini-servidor Python que receberá os arquivos para avaliação do módulo “Apresentação”, realizará a avaliação e retornará o resultado ao módulo “Apresentação”.

6.3.1 Módulo “Apresentação”

O módulo de “Apresentação” é responsável por receber o projeto do usuário por meio da interface web, enviar o projeto ao módulo “Appsthetics” (Figura 50) e apresentar o resultado da avaliação ao usuário.

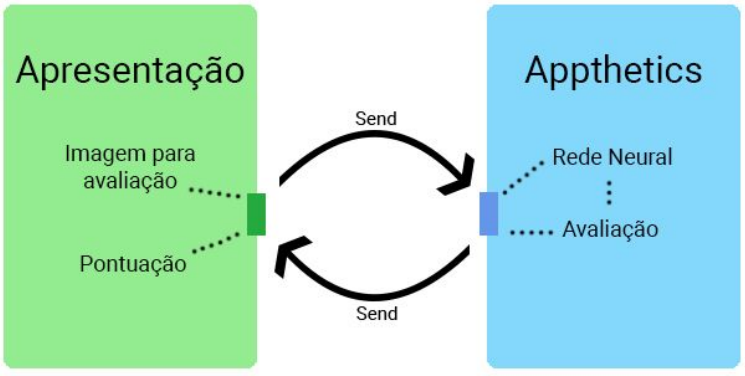

Figura 50: Comunicação entre os módulos (elaborado pelo autor)

O módulo “Apresentação” receberá do Aluno um ou mais arquivos no formato .jpg e fará o tratamento e envio das imagens para o módulo “Appsthetics” pelo método `doPost` da

classe `UploadAluno`. Um novo método `sendImageAndEvaluate` (Figura 51) foi criado nesta mesma classe, e também na classe `HTTPClient` (Figura 52), para tratar especificamente do envio da imagem para o módulo “Appsthetics” por uma requisição HTTP POST e o recebimento e retorno da pontuação obtida na predição do modelo.

```
private float sendImageAndEvaluate(File image) {
    float score = -1.0f;
    HTTPClient client = new HTTPClient();
    score = Float.parseFloat(client.sendImageAndEvaluate(image));
    return score;
}
```

Figura 51: Trecho do método sendImageAndEvaluate em UploadAluno.java (elaborado pelo autor)

```
public String sendImageAndEvaluate(File image) {
    String jsonScore = "";
    try {
        CloseableHttpClient httpclient = HttpClients.createDefault();
        HttpPost httppost = new HttpPost("http://localhost:9999/");
        FileBody fileBody = new FileBody(image);
        MultipartEntity multipart = new MultipartEntity();
        multipart.addPart("file", fileBody);
        httppost.setEntity(multipart);
        CloseableHttpResponse response = httpclient.execute(httppost);
        jsonScore = EntityUtils.toString(response.getEntity());
        httpclient.close();
    } catch (IOException | ParseException ex) {
        log.error("Não foi possível enviar o projeto para o RESTGrader ou receber a avaliação.", ex);
    }
    return jsonScore.replaceAll("^\"|\"$", "");
}
```

Figura 52: Método sendImageAndEvaluate em HTTPClient.java (elaborado pelo autor)

6.3.2 Módulo “Appsthetics”

O módulo “Appsthetics” consiste em um mini-servidor Python utilizando do pacote `socket` para realizar uma conexão direta com o módulo “Apresentação”. O servidor escuta pela porta 9999, como definido pelos requisitos, e estabelece uma conexão ao receber uma requisição do módulo “Apresentação”, recebe os dados da imagem, em formato hexadecimal, em duas partes: primeiro seu tamanho e então o próprio conteúdo da imagem. O módulo “Appsthetics” então reconstrói a imagem a partir destes dados, realiza a avaliação com a rede neural treinada e retorna a pontuação obtida como resultado para o módulo “Apresentação” (Figura 50). O código completo do módulo “Appsthetics” encontra-se no Apêndice B.

```python
@app.route("/", methods=['POST'])
def home():
    data = request.files['file']
    print(request.files['file'])

    appsthetics = predictor.Predictor()
    score = str(appsthetics.predict_aesthetic(data)[0])[1:-1]
    if float(score) < 0.0:
        score = str(0.0)
    print(score)

    response = jsonify(score)
    response.headers.add('Access-Control-Allow-Origin', '*')

    return response
```

Figura 53: Trecho do mini-servidor Python (elaborado pelo autor)

A Figura 54 mostra a classe `Predictor`, onde a rede neural é carregada e executada. `abs_path` é uma variável auxiliar para referenciar o diretório local, enquanto `dname` recebe o nome do diretório local. O comando `os.chdir(dname)` serve para trocar o contexto de execução para o diretório local, onde o arquivo da rede treinada (`export.pkl`) se encontra. A rede é então carregada com o método `load_learner` do pacote `fastai.vision`.

```python
from fastai.vision import *
from tkinter import filedialog
import os

abspath = os.path.abspath(__file__)
dname = os.path.dirname(abspath)
os.chdir(dname)


class Predictor:

    def __init__(self):
        base_path = Path(dname)
        self.learn = load_learner(base_path)

    def predict_aesthetic(self, file):
        img_cnn = open_image(file)
        predicted_score = self.learn.predict(img_cnn)
        return predicted_score
```

Figura 54: predictor.py (elaborado pelo autor)

O método `predict_aesthetic` é então definido:

- Recebe um arquivo como entrada.
- O método `open_image` abre a imagem e armazena os dados em `img_cnn`.
- O método `predict` da rede é chamado para gerar a predição e o resultado é

armazenado em `predicted_score` e então retornado.

6.3.3 Adaptações de Interface no CodeMaster

Conforme a integração da nova funcionalidade proposta, ajustes nas telas foram feitas. A seguir são apresentadas as principais alterações seguindo o fluxo do Caso de Uso 1.

Na página “Aluno” onde é feito o upload de arquivos por parte dos Alunos foi introduzido um novo item referente à opção de escolher uma imagem para *upload* (Figura 55). Também são informados os requisitos para o aceite da imagem.

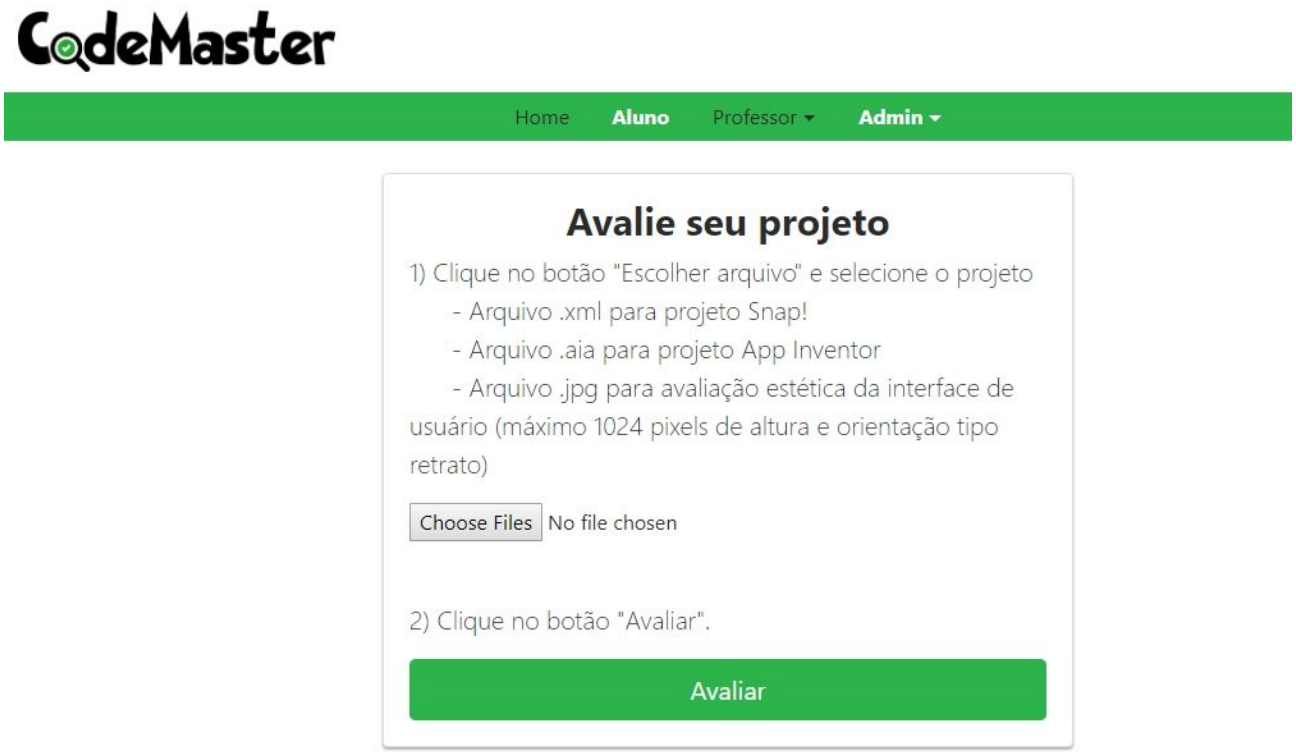

Figura 55: Página “Aluno” (elaborado pelo autor)

Ao clicar em “Choose Files” agora também são exibidas imagens em formato .jpg, que podem ser selecionadas pelo Aluno (Figura 56). Múltiplas imagens podem ser selecionadas ao mesmo tempo.

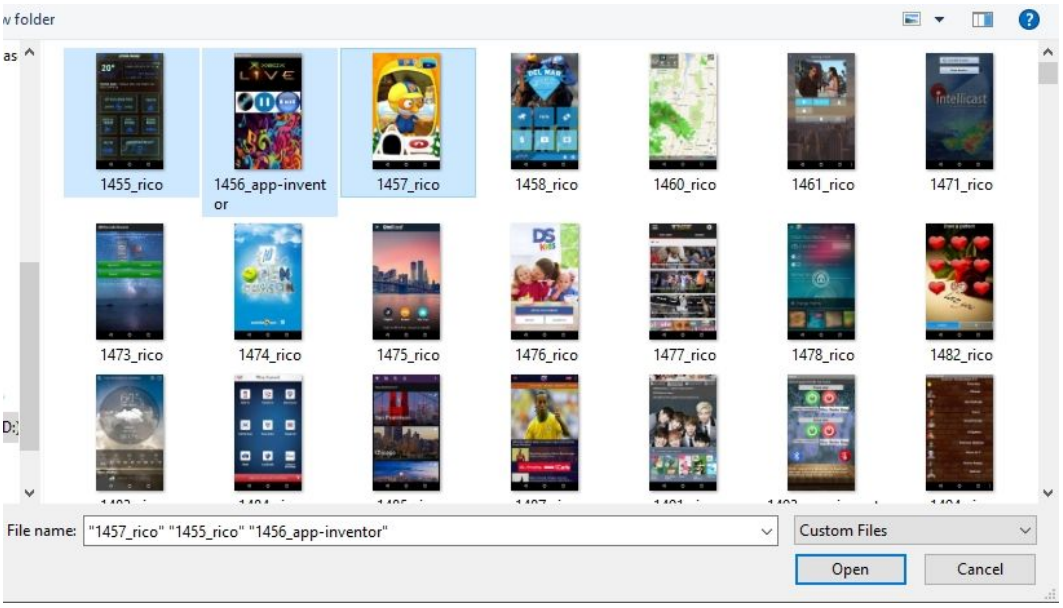

Figura 56: Seleção de múltiplos arquivos formato .jpg (elaborado pelo autor)

Após a seleção de uma imagem o Aluno clica em “Avaliar” e, caso todos os requisitos tenham sido respeitados, o Aluno é redirecionado para a página resultado, que agora contém uma nova tabela na aba “Interface de Usuário” onde é exibida a imagem avaliada e a pontuação obtida pela predição do modelo “Appsthetics” (Figura 57). O Aluno também possui a opção de realizar novas avaliações por meio do botão disponibilizado.

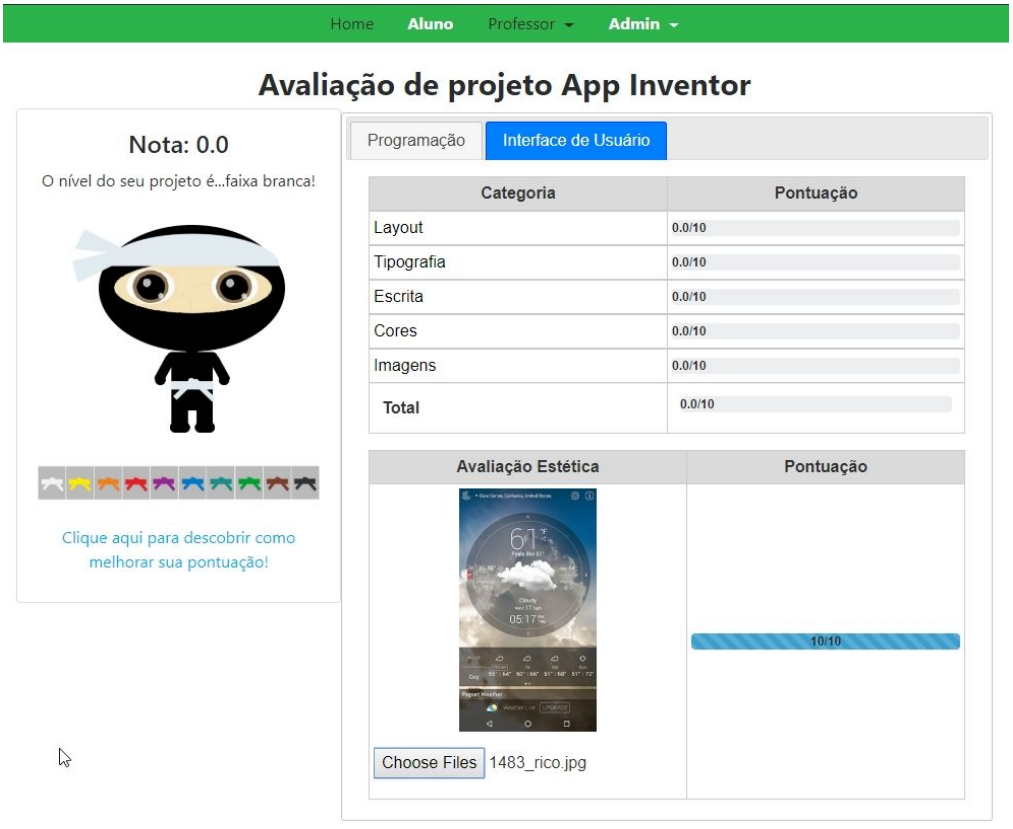

Figura 57: Nova área de estética na aba Interface de Usuário (elaborado pelo autor)

O sistema CodeMaster se encontra disponível em http://apps.computacaonaescola.ufsc.br:8080, enquanto o mini-servidor Python contendo o modelo Appsthetics se encontra na porta 9999.

7. CONCLUSÃO

O objetivo principal do presente trabalho foi de desenvolver um modelo de avaliação da estética visual da interface de usuário de aplicativos Android utilizando *deep learning* e integrar este modelo na ferramenta CodeMaster viabilizando a sua utilização no ensino de computação na Educação Básica. Inicialmente foi realizada a análise da fundamentação teórica sobre a estética visual de interfaces de usuários e também sobre inteligência artificial no contexto de técnicas de *deep learning* (O1). Também foi levantado o estado da arte em relação a análise automática da estética visual de interfaces de usuário (O2). A análise revelou diversos avanços em avaliações da estética em áreas como fotografias e *websites*, contudo identificou-se uma ausência de modelos e ferramentas específicos para aplicativos de smartphone. O modelo de rede neural "Appsthetics" foi desenvolvido e testado com técnicas de *deep learning* e com o auxílio da biblioteca Fastai (Fastai, 2019) (O3) como tarefa de regressão. Para assegurar a corretude do modelo os resultados dos testes foram analisados tanto com o coeficiente de correlação de Pearson (RUSSEL, 2015) como com a análise de Bland-Altman (BLAND, ALTMAN, 1986). Observando o desempenho satisfatório do modelo (ResNet50), também foi desenvolvida a implementação da avaliação automatizada da estética visual de interfaces de usuário incluída no sistema CodeMaster (O4).

Em termos de impacto científico, é criado um conhecimento pioneiro e inovador em relação a avaliação da estética visual de interfaces de usuário de aplicativos Android. É criado também, como resultado tecnológico, um modelo inovador que permite a análise automatizada da estética visual de interfaces de usuário de apps. Em termos sociais, com a disponibilidade do modelo desenvolvido, espera-se contribuir com a melhoria do ensino de computação no âmbito do ensino Brasileiro com relação à aprendizagem e popularização da computação e especialmente o design visual na sociedade.

Como trabalhos futuros sugere-se realizar uma análise mais ampla do conjunto de dados e do próprio modelo, visando melhorar e adaptar ambos de acordo com os erros atualmente observados. Recomenda-se também desenvolver uma opção de avaliação do próprio código-fonte da interface dos apps para obter um nível de análise mais profundo.

REFERÊNCIAS BIBLIOGRÁFICAS

ALVES, N. da C., GRESSE VON WANGENHEIM, C., HAUCK, J. C. R. **Um Modelo de Avaliação do Pensamento Computacional na Educação Básica através da Análise de Código de Linguagem de Programação Visual**. Anais do VIII Congresso Brasileiro de Informática na Educação. Brasília/DF, 2019.

APOSTOLIDIS, K.; MEZARIS, V.; **Image Aesthetics Assessment Using Fully Convolutional Neural Networks**. MultiMedia Modeling (MMM), Lecture Notes in Computer Science, vol. 11295, 2019, pp 361-373.

BABICH, N.; A Comprehensive Guide To Mobile App Design, 2018. Disponível em: <https://www.smashingmagazine.com/2018/02/comprehensive-guide-to-mobile-app-design/>. Acesso em: Abril 2019.

BLAND, J. M.; ALTMAN, D. G.; **Statistical methods for assessing agreement between two methods of clinical measurement**. Lancet, edição 8476, vol. 327, 2019, pp 307-310.

BRIGGS, D.; **The Dimensions of Colour**. MODERN COLOUR THEORY FOR TRADITIONAL AND DIGITAL PAINTING MEDIA. Sydney, Australia, Julian Ashton Art School and National Art School. Disponível em: <http://www.huevaluechroma.com/011.php>. Acesso em: Abril 2019.

BURNETTE, C.; **The Role of Aesthetics in Design Thinking**, 2015. Disponível em: <https://www.academia.edu/19251847/The_Role_of_Aesthetics_in_Design_Thinking>. Acesso em: Abril 2019.

BUSHAEV, V.; **How do we 'train' neural networks?**, 2017. Disponível em: <https://towardsdatascience.com/how-do-we-train-neural-networks-edd985562b73>. Acesso em: Junho 2019.

CORDEIRO, Alexander Magno et al . Revisão sistemática: uma revisão narrativa. **Revista do Colégio Brasileiro de Cirurgiões**, vol. 34, n. 6, p. 428-431, 2007.

CORTES, C.; VAPNIK, V.; **Machine Learning**. Kluwer Academic Publishers-Plenum Publishers, 1995, vol. 20, no. 3, pp 273-297.

CSTA. ACM. **CSTA K –12 Computer Science Standards**, 2016. Disponível em: <http://k12cs.org/wp-content/uploads/2016/09/K–12-Computer-Science-Framework.pdf>. Acesso em: Outubro 2018.

DEMETRIO, M. F. **Desenvolvimento de um analisador e avaliador de código de App Inventor para ensino de computação**. Trabalho de Conclusão do Curso de Bacharel em Ciências da Computação da Universidade Federal de Santa Catarina, Florianópolis, Brasil, 2017.

DONDIS, D. A.; **Primer of Visual Literacy**. Cambridge, MA: The MIT Press, 1973.

Fastai; **About**. Disponível em: <https://www.fast.ai/about/>. Acesso em: Maio 2019.

FERREIRA, N. F.; **Design de Interfaces para apps: proposição de uma unidade de ensino para estudantes do ensino fundamental II**. Dissertação de mestrado, Programa de Pós-Graduação em Design – PósDESIGN/UFSC, Florianópolis, Brasil, 2019.

FONNEGRA, R. D.; BLAIR B.; DÍAZ, G. M; **Performance Comparison of deep learning frameworks in image classification problems using convolutional and recurrent networks**. IEEE Colombian Conference on Communications and Computing (COLCOM), Cartagena, Colômbia, 2017.

FOWLER, M.; **Inversion of Control**, 2005. Disponível em: <https://martinfowler.com/bliki/InversionOfControl.html>. Acesso em: Maio 2019.

FUMO, D.; **Types of Machine Learning Algorithms You Should Know**, 2017. Disponível em: <https://towardsdatascience.com/types-of-machine-learning-algorithms-you-should-know-953a08248861>. Acesso em: Maio 2019.

GALITZ, W. O.; **The Essential Guide to User Interface Design: An Introduction to GUI Design Principles and Techniques**. (3ª edição) Hoboken, NJ: John Wiley & Sons, 2007.

GIAVARINA, D.; **Understanding Bland Altman analysis**. Biochemia Medica, vol. 25, no. 2, 2015, pp. 141-151.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A.; **Deep Learning**. Cambridge, MA: MIT Press, 2016.

Google, Google. Disponível em: <https://www.google.com>. Acesso em: Abril 2019b.
Google, Machine Learning: Practica - Image Classification. Disponível em: <https://developers.google.com/machine-learning/practica/image-classification/convolutional-neural-networks>. Acesso em: Outubro 2018.

Google, Material Design - **Icons**. Disponível em: <https://material.io/tools/icons/>. Acesso em: Abril 2019e.

Google, Material Design - **Responsive layout grid**. Disponível em: <https://material.io/design/layout/responsive-layout-grid.html>. Acesso em: Abril 2019a.

Google, Material Design - **The color system** - 2014 Material Design color palettes. Disponível em: <https://material.io/design/color/the-color-system.html#tools-for-picking-colors>. Acesso em: Abril 2019d.

Google, Material Design - **The type system**. Disponível em: <https://material.io/design/typography/> . Acesso em: Abril 2019c.

GRESSE VON WANGENHEIM, C.; ARAUJO PORTO, J. V.; HAUCK, J. C. R.; BORGATTO, A. F.; **Do we agree on user interface aesthetics of Android apps?** arXiv:1812.09049 [cs.SE], 2018b.

GRESSE VON WANGENHEIM, C.; HAUCK, J. C. R.; DEMETRIO, M. F.; PELLE, R. ALVES, N. d. C.; BARBOSA, H.; AZEVEDO, L. F. **CodeMaster – Automatic Assessment and Grading of App Inventor and Snap! Programs**. Informatics in Education, vol. 17, n. 1, 2018a, 117-150.

HATTIE, J.; TIMPERLEY, H.; **The Power of Feedback**. Review of Educational Research, vol. 77, n. 1, 2007, p. 81-112.

HE, K.; ZHANG, X.; REN, S.; SUN, J.; **Deep Residual Learning for Image Recognition**. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, Estados Unidos, 2016, pp 770-778.

HOU, K.C.; HO, C.H.; **A Preliminary Study on Aesthetics of Apps Icon Design**. Proceedings of 5th International Congress of the International Association of Societies of Design Research, Tóquio, Japão, 2013.

HOWARD, J.; **fastai v1 for PyTorch: Fast and accurate neural nets using modern best practices**, 2018. Disponível em: <https://www.fast.ai/2018/10/02/fastai-ai/>. Acesso em: Maio 2019.

ImageNet. **About ImageNet**. Disponível em: <http://image-net.org/about-overview>. Acesso em: Junho 2019.

INDIA, U.; **Difference between Machine Learning, Deep Learning and Artificial Intelligence**, 2018. Disponível em: <https://medium.com/@UdacityINDIA/difference-between-machine-learning-deep-learning-and-artificial-intelligence-e9073d43a4c3>. Acesso em: Maio 2019.

ITTEN, J.; **The Art of Color**: The Subjective Experience and Objective Rationale of Color. Hoboken, NJ: John Wiley & Sons, 1997.

JERSEY; **RESTful Web Services in Java**. 2017. Disponível em: <https://eclipse-ee4j.github.io/jersey.github.io/>. Acesso em: Outubro 2019.

KAHIRI, A.; **Adversarial Training**: Creating Real Pictures of Fake People With Machine Learning, 2018. Disponível em: <https://towardsdatascience.com/adversarial-training-creating-realistic-fakes-with-machine-learning-c570881d0e81>. Acesso em: Maio 2019.

KAIRANBAY, M.; SEE, J.; WONG, L. K.; **Aesthetic Evaluation of Facial Portraits Using Compositional Augmentation for Deep CNNs**. Proceedings of 13th Asian Conference on Computer Vision (ACCV), Taipei, Taiwan, 2016, pp 462-474.

KARAYEV, S.; **Vislab**. Disponível em: <http://vislab.berkeleyvision.org>. Acesso em: Junho 2019.

KHOSLA, A.; RAJU, A. G.; TORRALBA, A.; OLIVA, A.; **Understanding and Predicting Image Memorability at a Large Scale**. Proceedings of the IEEE International Conference on Computer Vision, Washington, DC, Estados Unidos, 2015, pp 2390-2398.

KIERSKI, M. **Machine Learning development process - you've got it wrong**, 2017. Disponível em: <https://medium.com/sigmoidal/machine-learning-development-process-youve-got-it-wrong-396270e653f4>. Acesso em: Novembro 2018.

KONIOR, L.; MARTYNOWSKI, A.; **These 5 major UI mistakes will kill your app**, 2016. Disponível em: <https://www.invisionapp.com/inside-design/app-ui-design-mistakes/> Acesso em: Abril 2019.

LIDWELL, W.; HOLDEN, K.; BUTLER, J.; **Universal Principles of Design**. (2ª edição) Beverly, MA, Estados Unidos, Rockport Publishers, 2009.

LU, X.; LIN, Z.; JIN, H.; YANG, J.; WANG, J. Z. **Rating Image Aesthetics Using Deep Learning**. IEEE Transactions on Multimedia, vol. 17, no. 11, 2015, pp. 2021-2034.

LUNDGREN, A.; **What is Design Theory?**, 2018. Disponível em: <https://alvalyn.com/what-is-design-theory/>. Acesso em: Junho 2019.

MALU, G.; SURAMPUDI, B.; INDURKHYA, B. (2017). **Learning Photography Aesthetics with Deep CNNs**. Proceedings of the 28th Modern Artificial Intelligence and Cognitive Science Conference 2017, Fort Wayne, Estados Unidos, pp. 129-136.

MARCUS, A.; **Graphic Design for Electronic Documents and User Interfaces**. (1ª edição) Nova Iorque, NY:ACM Press, 1992.

MATSUGU, M.; MORI K.; MITARI, Y.; KANEDA, Y. **Subject independent facial expression recognition with robust face detection using a convolutional neural network**. Neural Networks, vol. 16, n. 5, p. 555–559, 2003.

MINIUKOVICH, A.; ANGELI A. D.. **Computation of Interface Aesthetics**. In Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems. ACM, Nova Iorque, NY, Estados Unidos, p. 1163-1172.

MIT App Inventor, About Us, 2012. Disponível em: <http://appinventor.mit.edu/explore/about-us.html>. Acesso em: Outubro 2018

MITCHELL, T. M; **Machine Learning**, 1ª edição, Nova Iorque, NY: McGraw-Hill Education, 1997.

NIKOLOV, A.; **Design principle: Aesthetics**, 2017. Disponível em: <https://uxdesign.cc/design-principle-aesthetics-af926f8f86fe>. Acesso em: Junho 2019.

NORMAN, D. A.; **Emotional Design**: Why We Love (or Hate) Everyday Things. Nova Iorque, NY: Basic Books, 2005.

PAJUSALU, M.; **The Evaluation of User Interface Aesthetics**. Dissertação de Mestrado. Universidade de Talín, Talín, Estônia, 2012.

PECK, R.; OLSEN, C; SHORT, T. **Introduction to Statistics and Data Analysis**. 8ª edição. Boston, MA: Cengage Learning, 2019.

POLYZOTIS, N.; ROY, S.; WHANG, S. E.; ZIENKEVICH, M. **Data Management in Production Machine Learning**. Proceedings of the ACM International Conference on Management of Data, Chicago, IL, Estados Unidos, 2017.

PRESSMAN, R. **Engenharia de software: Uma abordagem profissional**. 8ª edição. Porto Alegre: Bookman, 2016. 968 p.

REINECKE, K.; GAJOS, K. Z.; **Quantifying visual preferences around the world**. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. Nova Iorque, NY, Estados Unidos, 2014, pp 11-20.

RODRIGUEZ, A.; **Restful web services: The basics**. IBM developerWorks, 2008.

RUSSAKOVSKY, O.; DENG, J.; SU, H.; KRAUSE, J.; SATHEESH, S.; MA, S.; HUANG, Z.; KARPATHY, A.; KHOSLA, A.; BERNSTEIN, M.; BERG, A. C.; FEI-FEI, L.; **ImageNet Large Scale Visual Recognition Challenge**. International Journal of Computer Vision, vol. 115, no. 3, 2015, pp 211-252.

RUSSEL, S.; **Artificial Intelligence: A Modern Approach**, 3ª edição. Chennai, TN, India, Pearson Education India, 2015.

SCHLATTER, T.; LEVINSON, D.; **Visual Usability: Principles and Practices for Designing Digital Applications**. 1ª edição, Burlington, MA:Morgan Kaufmann Publishers, 2013.

SHAIKH, F.; **Top 5 Interesting Applications of GANs for Every Machine Learning Enthusiast!**, 2019. Disponível em: <https://www.analyticsvidhya.com/blog/2019/04/top-5-interesting-applications-gans-deep-learning/>. Acesso em: Maio 2019.

SHELLEY, J. **The Concept of the Aesthetic**. The Stanford Encyclopedia of Philosophy (Winter 2017 Edition), 2017, Edward N. Zalta.

SHUAI. **Steps to Follow in Deep Learning Projects**, 2017. Disponível em: <https://shuaiw.github.io/2017/09/27/steps-to-follow-in-deep-learning-projects.html>. Acesso em: Novembro 2018.

Skymind, **A Beginner's Guide to Convolutional Neural Networks (CNNs)**. Disponível em: <https://skymind.ai/wiki/convolutional-network>. Acesso em: Maio 2019b.

Skymind, **A Beginner's Guide to Generative Adversarial Networks (GANs)**. Disponível em: <https://skymind.ai/wiki/generative-adversarial-network-gan>. Acesso em: Maio 2019c.

Skymind, **A Beginner's Guide to Neural Networks and Deep Learning**. Disponível em: <https://skymind.ai/wiki/neural-network>. Acesso em: Maio 2019a.

SMITH, A.; Guidelines on How to Make a Great Mobile App Screen Design, 2018a. Disponível em:

<https://blog.prototypr.io/guidelines-on-how-to-make-a-great-mobile-app-screen-design-df1287738f25>. Acesso em: Abril 2019.

SMITH, L. N.; **A disciplined approach to neural network hyper-parameters: Part 1 - learning rate, batch size, momentum, and weight decay**, arXiv:1803.09820, 2018b.

SMITH, L. N.; TOPIN, N.; **Super-Convergence: Very Fast Training of Residual Networks Using Large Learning Rates**, arXiv:1708.07120, 2018.

SOLECKI, I.; PORTO, J. A.; JUSTEN, K. A.; ALVES, N. d. C., GRESSE VON WANGENHEIM, C., BORGATTO, A. F.; HAUCK, J. C. R. **CodeMaster UI Design – App Inventor: Uma Rubrica de Avaliação do Design de Interface de Aplicativos Android desenvolvidos com App Inventor**. Proceedings of the XVII Simpósio Brasileiro sobre Fatores Humanos em Sistemas Computacionais, Vitória, Brasil, 2019.

SRINIVASA, S.; **How visual perception affects visual design**. Disponível em: <https://uxdesign.cc/how-visual-perception-affects-visual-design-e0a845876d7b>. Acesso em: Junho 2019.

THUERING, M.; MAHLKE, S.; **Usability, aesthetics and emotions in human-technology interaction**. International Journal of Psychology, vol. 42, no. 4, 2007, pp 253-264.

TSENG, F.; KOGAN, G.; REFSGAARD, A.; **How neural networks are trained**, 2019. Disponível em: <https://ml4a.github.io/ml4a/how_neural_networks_are_trained/>. Acesso em: Junho 2019.

ULRICH, K. T.; **Aesthetics in Design**. Design: Creation of Artifacts in Society, 2006. Disponível em: <http://opim.wharton.upenn.edu/~ulrich/documents/ulrich-aesthetics.pdf>. Acesso em: Abril 2019.

VON WANGENHEIM, A.; **Visão Computacional**:Deep Learning, 2018. Disponível em: <http://www.lapix.ufsc.br/ensino/visao-computacional/visao-computacionaldeep-learning>. Acesso em: Junho 2019.

WAZLAWICK, R. S.; **Análise e Design Orientados a Objetos para Sistemas de Informação**: Modelagem com UML, OCL e IFML. 3ª edição, Rio de Janeiro: editora Campus, 2016.

WEISSTEIN, E.; **Convolution**. MathWorld -- A Wolfram Web Resource. Disponível em: <http://mathworld.wolfram.com/Convolution.html>. Acesso em: Maio 2019.

WIKIPEDIA; **Overfitting**. Disponível em: <https://en.wikipedia.org/wiki/Overfitting>. Acesso em: Junho 2019.

YOSINSKI, J.; CLUNE, J.; BENGIO, Y.; LIPSON, H.; **How transferable are features in deep neural networks?**, Proceedings of the 27th International Conference on Neural Information Processing Systems, Montreal, Canada, 2014, pp 3320-3328.

ZANGWILL, N.; **Aesthetic Judgement**, The Stanford Encyclopedia of Philosophy (Spring 2019 Edition), 2019, Edward N. Zalta.

ZEN, M.; VANDERDONCKT, J.; **Assessing User Interface Aesthetics Based on the Inter-Subjectivity of Judgement**. Proceedings of the 30th International BCS Human Computer Interaction Conference: Fusion!, Poole, Reino Unido, 2016, artigo n. 25.

ZHOU, B.; LAPEDRIZA, A.; KHOSLA, A.; OLIVA, A.; TORRALBA, A.; **A 10 million Image Database for Scene Recognition**. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 40, no. 6, 2018, pp 1452 - 1464.

APÊNDICE A

Para separar o conjunto de dados em um conjunto de treino e outro de testes foi criado um pequeno *script* em Python que pode ser visto na Figura 1.

```
import csv
import os
import shutil

path = r"D:/Files/Mine/UFSC/TCC/datasets/aesthetics_dataset_csv/train/"
test_set = r"D:/Files/Mine/UFSC/TCC/datasets/aesthetics_dataset_csv/test/"

scores = open(r"D:\Files\Mine\UFSC\TCC\datasets\aesthetics_dataset_csv\full_no_ads_labels.csv",
newline='')
reader = csv.DictReader(scores)

score_list = ['1.00', '0.83', '0.67', '0.50', '0.33', '0.17', '0.00']

count = 0
total = 0
for i in score_list:
   for item in os.listdir(path):
       scores.seek(0)
       for row in reader:
           if item == row['image'] + '.jpg':
               if row['score'] == i:
                   shutil.copy(path + row['image'] + '.jpg', test_set)
                   os.remove(path + item)
                   count += 1
                   print(i + ": " + str(count))
       if count == 90:
           total = total + count
           print(total)
           count = 0
           break
   if total == 630:
       break
```

Figura 1: *Script* Python para separação do conjunto de dados (elaborado pelo autor)

APÊNDICE B

O código completo do mini-servidor utilizado para o *deployment* da rede neural Appsthetics junto ao sistema web CodeMaster é disponibilizado na Figura 1.

```
from flask import Flask, request, jsonify
import os
import predictor

app = Flask(__name__)


@app.route("/", methods=['POST'])
def home():
    data = request.files['file']
    print(request.files['file'])

    appsthetics = predictor.Predictor()
    score = str(appsthetics.predict_aesthetic(data)[0])[1:-1]
    if float(score) < 0.0:
        score = str(0.0)
    print(score)

    response = jsonify(score)
    response.headers.add('Access-Control-Allow-Origin', '*')

    return response


if __name__ == "__main__":
    from waitress import serve

    serve(app, host=os.getenv('HOST', '127.0.0.1'),
          port=os.getenv('PORT', '9999'),
          connection_limit=os.getenv('CONNECTION_LIMIT', '1000'),
          threads=os.getenv('THREADS', '50'),
          channel_timeout=os.getenv('CHANNEL_TIMEOUT', '10'),
          cleanup_interval=os.getenv('CLEANUP_INTERVAL', '30'))
```

Figura 1: Código do mini-servidor Python usado no *deployment* (elaborado pelo autor)

APÊNDICE C

Automated Assessment of Visual Aesthetics of Android User Interfaces with Deep Learning

Adriano Lima[1], Osvaldo P. Heiderscheidt Roberge Martins[1], Christiane Gresse von Wangenheim[1], Aldo von Wangenheim[1], Adriano F. Borgatto[1], Jean C. R. Hauck[1]

[1]Department of Informatics and Statistics/Federal University of Santa Catarina, 88049-200 Florianópolis - SC, Brazil

adriano.lima@ufsc.br, martins.osvaldo@grad.ufsc.br, c.wangenheim@ufsc.br, awangenh@inf.ufsc.br, adriano.borgatto@ufsc.br, jean.hauck@ufsc.br

Abstract. Visual aesthetics are seen as an essential success factor of mobile applications affecting the users' experience and perception. Recently, machine learning approaches have shown great promise in being able to assess visual aesthetics, yet focusing on web user interfaces only. Therefore, we present a deep learning approach to automatically quantify the visual aesthetics of Android mobile user interfaces (MUIs) adopting a regression-based approach. We employ pairs of an Android app screen image and user rating to train a convolutional neural network (CNN) adopting a supervised machine learning approach. Our evaluation results demonstrate that a CNN can learn the prediction of visual aesthetics of MUIs based on the images of the screenshots with a mean squared error of 0.051369 using a test set of 630 screenshots. The predictions from the model are highly correlated with human ratings (Pearson correlation coefficient r = 0.74, p < 2.2e-16) and results of a Bland & Altman analysis indicate that more than 96% of them are in agreement with the human ratings. These promising results indicate that our model can serve as an effective and efficient means for assessing visual aesthetics during the user interface design process.

1. Introduction

Visual aesthetics are increasingly seen as an essential factor in perceived usability, interaction, and an overall appraisal of user interfaces (Ben-Bassat et al., 2006; Tuch et al. 2012; Hamborg et al., 2014; Silvennoinen, 2014). It represents an integral part of usability as a software product quality of user interfaces and refers to the beauty or the pleasing appearance of user interfaces of an interactive software system (Tractinsky et al., 2000; ISO/IEC 25010, 2011). Positive aesthetic responses often lead to positive interface interactions (Bhandari et al., 2019), giving the users the immediate and long-lasting impression that they are also useful and easier to use (Norman, 2002; Tuch et al., 2012; Tractinsky, 2006). Appealing aesthetics may even compensate for poor usability (Anderson, 2011; Zen and Vanderdonckt, 2016, Bhandari et al., 2019). Furthermore, the perceived visual aesthetics often determine if a system is going to be used or avoided in favor of competitive systems (Schenkman and Jönsson, 2000; Tractinsky, 2013; Lu et al., 2013). Visual aesthetics becomes even more important with respect to the success of mobile apps (Miniukovich and De Angeli, 2014b; Gupta and Kohli, 2015; Bhandari et al., 2017). With millions of apps to choose from, a first impression of the visual design is often the differentiating factor that becomes decisive in the choice of an app (Miniukovich and De Angeli, 2015; Bhandari et al., 2019).

Although, there exist already research investigating first impressions on desktop interfaces or web sites (Lindgaard et al., 2006), relatively little research has addressed mobile user interfaces (MUIs) (Miniukovich and De Angeli, 2014b; Bhandari et al., 2017). MUIs are essentially different from user interfaces on other devices as the graphical and touch-sensitive display allows the user to

interact with the device's apps, features, content, and functions in a different way (Litchfield, 2010). Clearly, the size and portability requirements of mobile phones present limitations as well as the awkward ways for input. And, although touch screen capabilities may facilitate certain actions, they also pose new challenges through the lack of tactile feedback, touch key size, etc. (Balagtas-Fernandez et al., 2009). Mobile devices also change traditional interaction models based on the familiar WIMP (Windows, Icons, Menus, Pointer) interface style to interactions that may involve voice, gestures, sensors and location data (Wasserman, 2010). Differences are also related to the environment, as mobile apps are being used anywhere anytime by a wide range of people with different goals (Huang, 2009). Smartphone users, however, might value visual aesthetics even more as the on-the-go usage implies multiple external distractions, and short and intensive interaction periods (Choi and Lee, 2012).

This indicates the importance of the design of visually attractive user interfaces as part of the software development process (Myers and Rosson, 1992; Zen and Vanderdonckt, 2014). Yet, it is typically a time-consuming activity, following guidelines and heuristics to design interface artifacts (Zheng et al., 2009), involving multiple iterations until all requirements are met (Martinez et al., 2017). And, given the importance of visual aesthetics, it is vital that it is adequately assessed (Moshagen and Thielsch, 2010). Yet, how to assess user interface aesthetics remains a question faced during the user interface development process (Zen and Vanderdonckt, 2016).

A common approach is to assess visual aesthetics as part of usability tests, with users rating de visual aesthetics of the interface. Yet, this is an expensive and time-consuming method demanding considerable resources that might be unavailable for small companies (Miniukovich and de Angeli, 2015). A way to minimize that effort is to automate the aesthetic assessment process. Diverse methods are applied to automate the assessment of aesthetics analyzing how design factors and layout elements influence users' perception of visual aesthetics (Moshagen and Thielsch, 2010; Zen and Vanderdonckt, 2016; Seckler et al., 2015; Kim et al., 2003). Approaches vary from the simple numerical count of interface elements to more complex algorithms that predict handcrafted features such as colorfulness and symmetry (Moshagen and Thielsch, 2010; Miniukovich and de Angeli, 2015; Seckler et al., 2015; Zen and Vanderdonckt, 2016) or even physiological measures such as facial electromyography and electrodermal responses (Bhandari et al., 2017). But, by examining design factors independently, these methods may not capture the complexity of the perception of aesthetics by users as a whole (Bhandari et al., 2019, Dou et al. 2019) and so far, there is no consensus on how to consistently assess the visual aesthetics of user interfaces (Zen and Vanderdonckt, 2016).

More recently, deep learning models are being applied to quantify the aesthetics of visual design of user interfaces such as webpages (Khani et al., 2016; Dou et al., 2019), due to the success on the evaluation of the aesthetics of photographs (Lu et al., 2014; Deng et al., 2017; Malu et al., 2017; Suchecki and Trzciski, 2017). These deep learning approaches are capable of automatically extracting high-level features directly from raw input data via a hierarchical architecture to map the perception of images to their aesthetics ratings (LeCun et al., 2015; Polyzotis et al., 2018). Image aesthetics is typically represented by discrete values, and the problem of assessing image aesthetics is formulated as either classification or regression (Lu et al., 2015). There exist deep learning approaches for the assessment of the aesthetics of desktop systems (Tractinsky, 2012) or web sites (Lindgaard et al., 2011; Reinecke et al., 2013) and how it affects user experience and usability based on feature modeling. Yet, considering the differences of mobile user interfaces (MUIs), so far research specifically focusing on the automatic assessment of the visual aesthetics of mobile user interfaces is basically nonexistent. Thus, the question that remains is how to assess the visual aesthetics of mobile user interfaces with an acceptable agreement with user ratings.

Therefore, we propose a deep learning approach *Appsthetics* to quantify the visual aesthetics of Android user interfaces with convolutional neural networks (CNN). We focus on Android apps due to the worldwide predominance of Android devices (IDC, 2019; Statcounter, 2019). The main contributions of our research are:

- Creation of a labeled dataset of 3,139 images of Android user interfaces publicly available at http://www.lapix.ufsc.br/mobile-interfaces-dataset.
- Development of a deep learning model to automatically assess the visual aesthetics of Android user interfaces with aesthetics ratings that are highly correlated with collected user rating data. The model is publicly available at https://codigos.ufsc.br/aldo.vw/appsthetics.

Enabling the automated assessment of the visual aesthetics of the MUIs can be beneficial to predict, ensure and increase the quality of Android apps, tackling some usability issues early, and therefore reducing development costs. It can further help to make systematic improvements to the development processes.

2. Related work

Previous research on visual aesthetics of interfaces has focused on the assessment of webpage interfaces on desktops. The focus of these research works has been on relating visual aesthetics to usability (Roast et al. 2002; Hartmann et al. 2007; Tuch et al. 2012), credibility (Robins and Holmes, 2008; Alsudani and Casey, 2009), user satisfaction (Tractinsky et al., 2000; Lindgaard and Dudek, 2002; Coursaris and van Osch, 2016), customer loyalty (Chang and Chen, 2009), or on how aesthetics evaluations are formed on users' first impressions (Lindgaard et al., 2006; Tractinsky et al., 2006).

These results have been extended to the mobile domain. Sonderegger and Sauer (2010) found that not only do users rate appealing mobile devices as more useful, but also demonstrate lower task completion times when operating those devices. Regarding mobile webpage interfaces, Oyibo et al. (2017) showed that aesthetics have a stronger influence on credibility than usability. Botzenhardt et al. (2016) highlighted aesthetics, as a key predictor of user satisfaction in the context of mobile banking services. Xu et al. (2015) concluded that aesthetics can have positive effects on user satisfaction, leading to a stronger intention to continue to use mobile apps. Kumar et al. (2018) showed that design aesthetics, perceived usefulness, ease of use and enjoyment have a significant influence on loyalty towards apps. And, Miniukovich and De Angeli (2014b) extending previous work on first impressions on interfaces evaluation (Lindgaard et al., 2006; Tractinsky et al., 2006) to MUIs, suggest that users form first impressions on mobile app interfaces the same way as with respect to widescreen user interfaces.

The assessment of the visual aesthetics of MUIs has also been extended from the way widescreen user interfaces are assessed. Much of the research has focused on handcrafted features. Miniukovich and De Angeli (2014b) confirmed that six metrics validated before for assessing web interfaces, including color depth, dominant colors, visual clutter, symmetry, figure-ground contrast, and edge congestion, can help to predict the visual quality of Android (Miniukovich and De Angeli, 2014a) as well as iPhone MUIs (Miniukovich and De Angeli, 2015). Other research analyzes the correlation between user interface complexity and user-perceived quality in Android applications, aiming at proposing guidelines for user interface complexity by mining available Android applications (Alemerien and Magel, 2014; Taba et al., 2014).

More recently, with advances in machine learning, deep learning approaches are being introduced to predict the aesthetics of software interfaces. Khani et al. (2016) and Dou et al. (2019) used machine learning to assess the visual aesthetics of web pages. Both employed neural networks (AlexNet and CaffeeNet) pre-trained for example with the ImageNet dataset. Khani et al. (2016) used a hybrid model integrating convolutional neural networks (CNNs) with classical machine learning, creating a neural network with 5 convolutional layers and 2 fully connected multilayer perceptrons. The network was trained with a dataset of 418 screenshots of web pages rated with respect to their visual aesthetics by volunteers. Using a Support Vector Machine with a Gaussian Radial Basis Function for classification, they report an error rate of 34.15%. Dou et al. (2019) also used a neural network with 5 convolutional layers and 2 fully connected multilayer perceptrons, but,

differently to Khani et al. (2016), they used a pure connectionist approach with one extra layer for max-pooling and one extra output layer for regression. They trained their network with 398 screenshots of webpages, rated by approx. 40,000 users online. For learning, a backpropagation algorithm was used with a reported error rate of 20.41%. Dou et al. (2019) suggest that the best way to treat the task of predicting visual aesthetics is regression rather than classification and they attribute the performance improvements compared to Khani et al. (2016) mainly to the training mode used and the change to regression.

In spite of the success of CNNs to predict aesthetic values of photographs or webpages, so far, no research on the application of CNNs to assess the visual aesthetics of MUIs has been found.

3. Research methodology

Based on the results of related work on different kinds of user interfaces, we develop a deep learning model following the machine learning process proposed by Ashmore et al. (2019) and Amershi et al. (2019) as illustrated in Fig. 1.

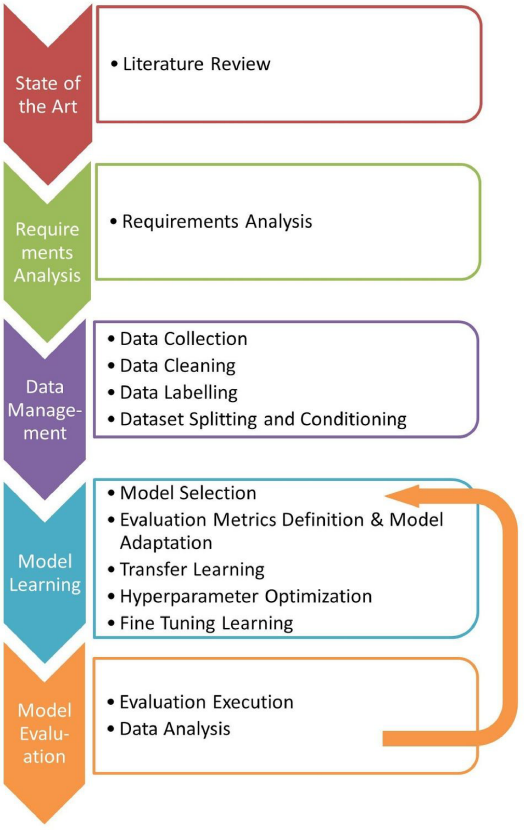

Fig. 1. Research methodology.

This process includes the following phases:

- **Requirements analysis**. During this phase, the main objective of the model and its target features is specified following Mitchell (1997). This also includes the characterization of the inputs and expected outputs, specifying the problem.
- **Data management**. During data collection, we searched for available datasets and collected additional data by capturing user interface screenshots. This includes the selection of available generic datasets for pre-training a model (e.g., ImageNet for image classification tasks), as well as specialized datasets for transfer learning to train the specific model. The data is prepared by validating and cleaning and conditioning the data (e.g., remove

duplicates, correct errors, deal with missing values, normalization, data type conversions, etc.). Data labeling assigns ground truth labels to each record. Adopting a supervised learning technique requires the manual assignment of labels by users. The dataset is split into a training set to train the model and a test set to perform an unbiased performance evaluation of the chosen model on unseen data (Ripley, 1996).

- **Model learning**. During the model learning phase, an appropriate deep learning framework and machine learning model has been chosen that has been proven effective for a comparable problem or domain, and the volume and structure of the data. Adopting a transfer learning approach, we take a pre-trained network, which was trained on a generic dataset (in our case, ImageNet), adapt its output layers in order to work with our evaluation metrics, and train the input layer and the final, fully connected output layers on our data, initially maintaining the pre-trained internal feature representation structure of the network. This is performed until the network does not improve anymore. After transfer learning, we unfreeze the internal features of the transfer-trained model allowing all its layers to learn. It then undergoes a training phase called fine-tuning, with the same dataset of the specific domain, where all internal features are finely adjusted to our data. A set of hyperparameters (HYPOs) (especially learning momentum and learning rates) for a learning algorithm is selected and dynamically optimized during the fine-tuning process. Here, we trained several similar model variants, with different HYPOs, and compared their results.
- **Model evaluation**. As soon as a model is trained, it is critical to evaluate its quality in order to understand how to iteratively improve its performance (e.g., in terms of high accuracy, lower error) by testing the model. Therefore, we defined the metric used for measuring success aligned with the goals to be achieved and the kind of problem faced by the model. Then, we tested the model(s) against previously unseen data (test set). In addition, we evaluated the model with respect to the correlation and agreement of the predicted scores to the ones given by human raters.

4. Appsthetics: a proposed deep learning model

Recent deep learning technologies, especially CNNs, have achieved broad successes on many extremely challenging image processing tasks (Dou et al., 2019). Thus, in order to support the automated assessment of the visual aesthetics of Android user interfaces we develop a deep CNN, called *Appsthetics*.

4.1 Requirements analysis

Following Mitchell (1997), our objective is to develop a computer program that learns from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E. Here, the Task (T) is to assess the degree of visual aesthetics of a MUI of an Android application on a numerical scale of [0..1], Experience (E) is a corpus of images of screenshots of Android applications with visual aesthetics varying from ugly to beautiful on a numerical scale [0..1], and Performance (P) is the mean squared error between the degree of visual aesthetic of a MUI predicted by the model *Appsthetics* and the human rating.

The input is represented by images of screenshots of Android applications (JPEG images with 239x408 pixels). The images are manually labeled by human raters, dealing with this problem by applying a supervised machine learning approach.

We propose to formulate the aesthetics assessment task as a real-valued regression problem rather than a classification problem following Dou et al. (2019). Thus, our goal is to predict continuous Android app screen aesthetics rating scores other than discrete category labels. Considering the task a regression problem aiming to predict continuous values with respect to the

visual aesthetics, the output is a numerical value [0..1] being interpreted as the degree of visual aesthetics with 0=ugly and 1= beautiful.

4.2 Data management

We created a new dataset for the assessment of the visual aesthetics of user interfaces of Android apps with aesthetics ratings.

Data collection. We collected 4500 images of Android user interface screenshots from different sources in October-December 2018:

- 4160 screenshots of Android app interfaces, taken from the RICO[17] dataset, one of the largest repositories containing over 72k unique UI screens of Android apps from the Google Play Store.
- 328 screenshots of Android app interfaces from the MIT App Inventor Gallery[18], being one of the most popular programming environments for teaching app development worldwide. These screenshots were semi-automatically acquired with a Samsung Galaxy S7 and a Sony Xperia XZ devices.
- 12 screenshots of Android app interfaces from apps developed by the Software Quality Group of the Federal University of Santa Catarina. These screenshots were semi-automatically acquired with a Samsung Galaxy S7 device.

Data cleaning/pre-processing. We eliminated all duplicates and selected only images that do not contain ads and that were deemed acceptable with respect to ethical concerns. During pre-processing, we eliminated the Android status bar located at the top of the screenshots in order to avoid unintended variations not directly related to the user interfaces themselves. We also downsampled all screenshots to 239x408 pixels JPEG images.

Data labeling. The screenshots were manually labeled with respect to aesthetics. Therefore, the images in this initial set were randomly divided into sets of 150 images. Volunteers of the Software Quality Group and the Hiperlab at the Federal University of Santa Catarina rated the screenshots in November, 2018-March 2019[19]. Twenty-nine (20 male and 9 female) participants took part in this study with a background in either computer science or design. Each person rated images of one or several of these subsets with respect to the perceived visual aesthetics on a 3-point ordinal scale with the response alternatives: beautiful/neutral/ugly. Each image was rated by 3 persons (with a total of 13,500 ratings). The rating has been performed via an online questionnaire answered remotely by the participants. Each questionnaire contained a set of 150 screenshots and the rating scale with respect to the aesthetics for each of the screenshots. Although no time constraint was imposed, in general, participants spent about 15 minutes responding each questionnaire.

Taking into consideration a typical only moderate interrater agreement on visual aesthetics of Android app interfaces (Gresse von Wangenheim et al., 2018), we considered for the labeled dataset only interfaces for which at least 2 raters assigned the same rating. This resulted in a labeled dataset with a total of 3,139 images. Then, we assigned numerical values to each class by converting the labels into a mean score (beautiful=1.0, neutral=0.5 and ugly=0.0) and calculated the mean score for each app interface considering the task at hand a real-valued regression problem rather than a classification problem, even though the user ratings were made in a discrete format. Fig. 2 shows the final distribution of all 3,139 images per mean aesthetics score. The mean rating of these scores is 0.4938 ± 0.3381. Due to the removal of interfaces with larger variations of rater agreement, fewer data points with a mean score of 0.5 are included in the dataset, yet, the distribution remains sufficiently balanced considering the complete dataset.

---

[17] http://interactionmining.org/rico

[18] http://appinventor.mit.edu/explore/

[19] This study has been approved by the Ethics Committee of the Federal University of Santa Catarina (no. 2.903.849).

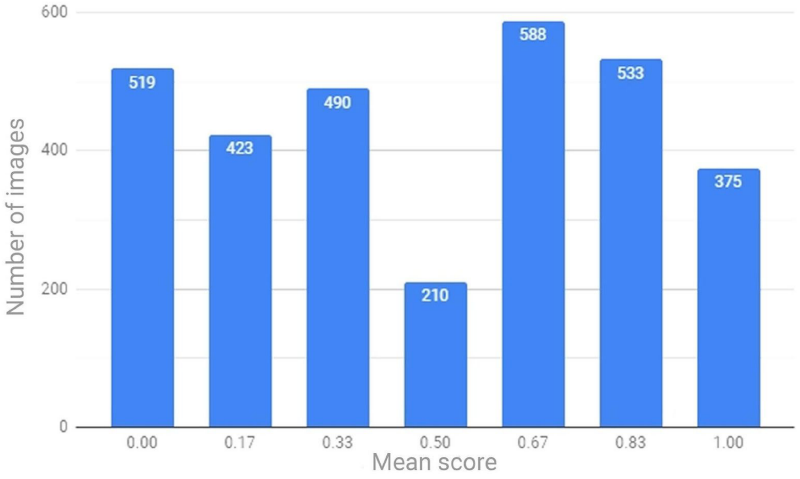

Fig. 2. Distribution of images per mean score.

Data splitting. We randomly divided the dataset into:

- Training set: a set of 2,509 images (80%) used for learning.
- Test set: A set of 630 images (20%), with 90 images of each mean score category.

The file with the evaluation data and the links to the images used are publicly available at http://www.lapix.ufsc.br/mobile-interfaces-dataset/.

4.3 Model learning

We employ pairs of an Android app screen image and user rating to train a CNN adopting a supervised machine learning approach. For the testing procedure, the ground truth of each Android app screen is the mean score of all user ratings it has received.

For the development of the machine learning model, we selected the fast.ai framework (Fast.ai, 2017). Fast.ai is a very high-level CNN framework based upon the PyTorch/Torch Python CNN framework, which is intended to make creating machine learning applications fast and simple (Howard, 2018). PyTorch/Torch framework has been considered to be one of the best performing and most flexible and research-oriented CNN framework available (Fonnegra et. al., 2017). Compared to lower-level libraries such as Keras, TensorFlow.Keras or pure PyTorch fast.ai reported reducing the amount of boilerplate code needed to produce state of the art neural network applications (Howard, 2018). The fast.ai library additionally provides advanced, yet, easy-to-use hyperparameter optimization strategies (HYPOs) that help in implementing better CNNs in a fast way (Howard, 2018). The rationale behind our design decision is based on the fact that we are performing research on the assessment of interface design employing deep learning techniques, rather than advancing the state of the art of deep learning technologies. In this case, a high-level framework, with readily available HYPOs poses the best conditions for the production of simple code and fast learning models.

Model selection. We opted for using residual networks (ResNets) (He et al., 2016), including the following network models: ResNet18, ResNet34, ResNet50, and ResNet101. Residual networks employ identity connections that act as shortcuts, bypassing several layers at a time, providing two parallel learning paths in several sections of the network and avoiding the gradient loss that typically occurs in very deep networks. This allows for much deeper networks with up to 152 layers. Their design principle also allows to choose or design a network with a specific depth, that is adequate for the complexity of the problem at hand. We chose residual networks (ResNets) mainly because they allow the use of HYPOs especially developed for these CNN models, which allow for faster training (Smith, 2018; Smith and Topin, 2018).

Adopting a transfer learning approach, we first train a base network on a base dataset towards a base task. When given a new target task, we initialize the layers of the target network with those of the base network and then fine-tune the whole target network on the target dataset. This transferability of learned knowledge increases as the distance between the target task and the base task decreases (Yosinski et al., 2014). Aiming to repurpose base models whose tasks share common ground with our target aesthetics assessment task, we pre-train the model with the ImageNet (Russakovsky et al., 2015), as one of the largest publicly available general-purpose data sets for the training of image classification as our base network.

Hyperparameter selection and optimization. We choose an optimization strategy for automatic hyperparameter selection (HYPO) called fit1cycle (Smith, 2018; Smith and Topin, 2018). The fit1cycle method was specifically developed for residual networks and works with varying, adaptive learning rates and momenta, following a curve where the rate is first automatically increased and then decreased and the momentum follows an opposed strategy (Smith and Topin, 2018). We trained all network models in two ways: employing this automatic hyperparameter selection strategy (automated hyperparameter optimization) as well as with fixed hyperparameters (standard training).

Transfer learning. As our dataset is relatively small, which may bring the risk of overfitting the model, we exploit transfer learning from a starting point with initialization from a pre-trained network. This strategy effectively enables us to train a deep network with limited data. Therefore, we adapted the input layer to the image resolution of our data set represented by a vector of an image of the user interface of an Android application and its label representing the mean score of user ratings for the user interface.

We also substituted the original output layers, which in the ImageNet pre-trained networks represent a categorical variable with 1000 values (containing 1000 neurons for ImageNet dataset) through a regression layer with a single-neuron. In addition, the classification cross-entropy loss is changed into regression loss. The result, given an input image of the user interface of an Android app, is, thus, a predicted aesthetic score as a real number ranging between 0.0 and 1.0.

The output layers of the ResNet18, ResNet34, ResNet50 and ResNet101 networks were transfer-trained until the test error did not show any improvement in relation to the training error. We trained our networks using the adaptation of the back-propagation algorithm developed for the ResNets (He et al., 2016).

Training. The fine-tuning phase was performed employing the same strategy as the transfer learning phase, only unfreezing and allowing for the adaptation of all weights in the network. We fine-tuned all networks also in two ways: using the fit1cycle strategy as well as without it. Table 1 summarizes the trained networks with both strategies and the results observed.

In both phases, transfer-learning and fine-tuning, training was performed using the following hyperparameters:

- **Standard training method** (*fit*): fixed learning rate of 0.003, no weight decay, no momentum management, and 10 epochs transfer-learning followed by 15 epochs fine-tuning.
- **Training with automated hyperparameter optimization** (*fit1cycle*): we employed the HYPO strategy developed by (Smith, 2018; Smith and Topin, 2018), using a global learning rate, always equal on all layers, global momenta varying in the range [0.85, 0.95], a learning rate division factor of 25 and no weight decay. The learning rates were always fixed at 0.003 for the transfer-learning phase. For the fine-tuning phase, we determined a range of optimum learning rates using the method suggested in (Smith and Topin, 2018). This resulted in different range of learning rates for each of the four networks. The values ranged between 10^{-5} and 10^{-7}, depending on the network. We provide these values in the source code.

Hyperparameters adaptation: The fit1cycle algorithm also operates in two phases: In phase 1, the learning rate varies linearly from the maximum learning rate divided by the division factor up to the maximum learning rate. At the same time, the momentum decreases linearly from the highest to the lowest value of the momenta range. In phase 2, the learning rate follows a cosine annealing from maximum learning rate to 0. At the same time momentum increases from the lowest value of the momenta range to its highest value, based upon the same cosine annealing. The fit1cycle policy allows to train very quickly, which is also called superconvergence (Smith and Topin, 2018).

Performance evaluation. The quality of a regression model is measured in terms of how well its predictions match up against actual values. The mean squared error (MSE) measures the average of the squares of the errors, that is, the average squared difference between the data labels (human rating score) and the value of the machine learning model. The MSE is always non-negative, values closer to zero are better.

Table 1. Summary of model training, strategies utilized and results.

Model	Training	Fine-tuning phase	Best mean squared error
ResNet18	Fixed hyperparameters	Unfrozen; Maximum learning rate limited for optimal learning	0.078778
	Hyperparameter optimization		**0.075770**
ResNet34	Fixed hyperparameters		0.078778
	Hyperparameter optimization		**0.065101**
ResNet50	Fixed hyperparameters		0.070096
	Hyperparameter optimization		**0.051369**
ResNet101	Fixed hyperparameters		0.082701
	Hyperparameter optimization		**0.052089**

Given the results presented in Table 1, we selected ResNet50 as it performed the best. The respective model is available at https://codigos.ufsc.br/aldo.vw/appsthetics.

5. Model evaluation

The aim of our research is to assess the visual aesthetics of Android MUIs that correspond with the real user ratings. Typically, correlation studies are used for this kind of evaluation. Thus, in order to compare our results to similar work, we perform a Pearson correlation analysis following the evaluation performed by Dou et al. (2019).

Correlation, however, quantifies the degree to which two variables are related, not the agreement between them. Therefore, correlation coefficient and regression techniques are sometimes inadequate and can be misleading when assessing agreement, because they evaluate only the linear association of two sets of observations (Giavarina, 2015). Thus, in order to assess the degree of agreement between automatic assessment of our model and the rating made by humans, we also use the Bland-Altman (B&A) plot analysis (Bland and Altman, 1986). The B&A plot analysis has been introduced to describe the agreement between two quantitative measurements by studying the mean difference and constructing limits of agreement. Therefore, it allows to evaluate a bias between the mean differences and to estimate an agreement interval, within which 95% of the differences of the second method, compared to the first one, fall.

5.1 Correlation analysis

In order to ensure that the trained model performs well on new, previously unseen inputs, the evaluation assesses the performance of the learned model against the test data set. We measure the strength of the linear association between the results of the deep learning network and ground truth based on human assessments through the Pearson correlation coefficient (r) (Bonett and Wright, 2000) that is computed as the ratio of covariance between the variables to the product of their standard deviations. The numerical value of r ranges from -1.0 to +1.0. The closer the coefficients are to +1.0 or -1.0, the greater the strength of the linear relationship is. We also calculated the 95% confidence intervals of the Pearson coefficients with the standard statistical method (Bonett and Wright, 2000). A p-value lower than 0.05 indicates statistical significance.

The results of the Pearson correlation analysis are shown in Fig. 3. The vertical lines in the plot reflect the mean values obtained from the human ratings for each interface on an ordinal 3-point scale. We can observe that the predicted values have a strong correlation with the human assessed ratings, with a coefficient $r = 0.74$, $p < 2.2e\text{-}16$. This means that the higher an interface is rated by the user, the higher is the predicted value by the model, reflecting that it has learned the features that represent the aesthetics of user interfaces of Android apps.

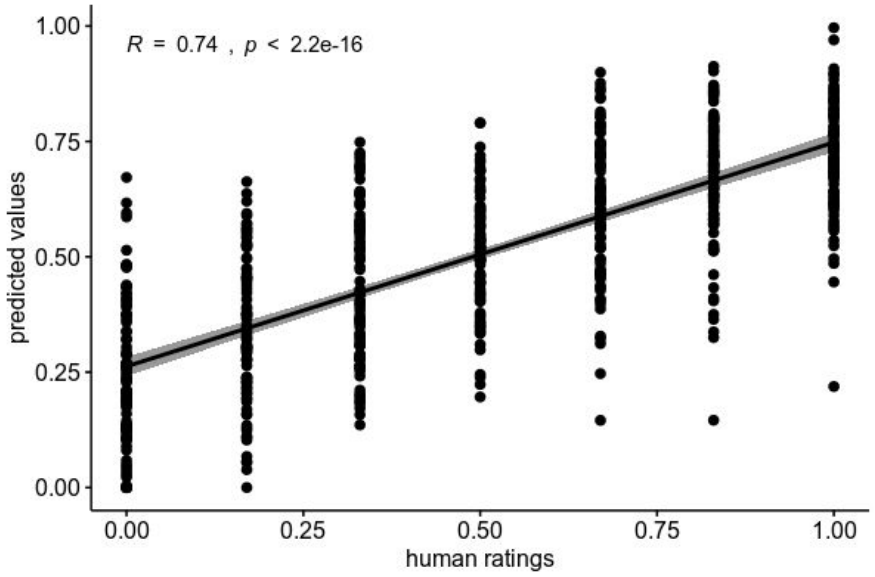

Fig. 3. Correlation plot (predicted values vs. human ratings).

Although considered a sufficient correlation, our results are slightly below the evaluation results reported by Dou et al. (2019) quantifying webpage aesthetics presenting a Pearson correlation of ($r = 0.85$, $p < .001$).

5.2. Bland-Altman analysis

We also analyzed the mean difference by quantifying the agreement between the results of the deep learning model (ResNet50) and ground truth based on human ratings with respect to the test set through the Bland-Altman plot analysis (Bland and Altman, 1986). This analysis does not indicate if the agreement between the predicted values and the human ratings is sufficient or if the automated assessment is suitable to replace the human one. It simply quantifies the bias and a range of agreement, within which 95% of the differences between one measurement and the other are included (Giavarina, 2015). B&A recommend that 95% of the data points should lie within ± 2s of the mean difference.

The B&A plot graph is shown in Fig. 4. Here, the human ratings are shown as diagonal lines. Although we can observe some variation on the difference between predicted values and human ratings, the analysis indicates that more than 96% of them agree, i.e., 605 out of the 630 outputs are within the 95% confidence interval. These numbers show a high agreement between the human ratings and the predicted values for the interfaces, with a bias close to zero (0.0051). Therefore, it is possible to say that the bias is not significant, as the line of equality (with the predicted and the assessed values being equal) is within the confidence interval of the mean difference (between -0.44 and 0.45).

As this type of analysis has not been performed by related work, no comparison is possible.

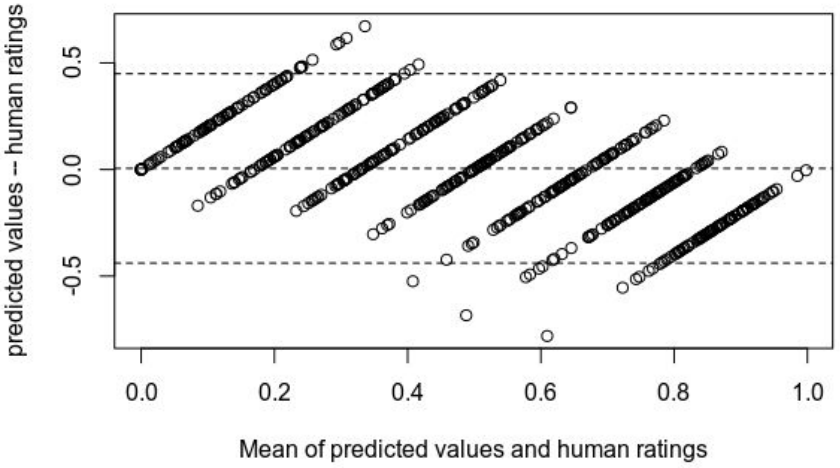

Fig. 4. Bland-Altman plot.

These evaluation results reflect that the factors that affect the visual aesthetics of Android user interfaces can be successfully represented by the features learned by the CNN model. Both the analysis of the Pearson correlation as well as the B&A analysis indicate a sufficient correlation/agreement between the results of the *Appsthetics* model and human ratings. And, although, the Pearson correlation is below the one reported by Dou et al. (2019) for the prediction of webpage aesthetics, it is still sufficient especially when considering the explorative nature of this research on MUIs.

6. Discussion

Automating the assessment of visual aesthetics of user interfaces is challenging as aesthetics may be rated differently by different people, and the optimal computational representation of aesthetics is not obvious. And although first approaches using deep learning have been explored for assessing the aesthetics of web pages, our proposed solution is novel by its focus on mobile user interfaces.

Our evaluation results demonstrate that a CNN can learn the prediction of visual aesthetics of MUIs based on the images of the screenshots. The predictions from the *Appsthetics* model are highly correlated with human ratings (Pearson correlation coefficient $r = 0.74$, $p < 2.2\text{e-}16$) and the Bland-Altman analysis indicates that more than 96% of them agree, i.e., 605 out of the 630 outputs are within the 95% confidence interval. We understand that these results demonstrate a sufficient accuracy considering the explorative nature of this research.

Although our results do not necessarily outperform related approaches to the assessment of websites (Dou et al., 2019) with respect to correlation, our approach is unprecedented for mobile user interfaces. Similar to Dou et al. (2019), we agree that the formulation of the task as regression

is a major factor in the performance achieved, as previous informal tests with classification models also yielded lower performance results.

Yet, in order to further understand the results, we examine some examples of assessment with high and low correspondence between the prediction of the *Appsthetics* model and the human ratings. Fig. 5 illustrates several specific examples of Android MUIs with both the human rating scores and the aesthetic scores predicted by the *Appthetics* model.

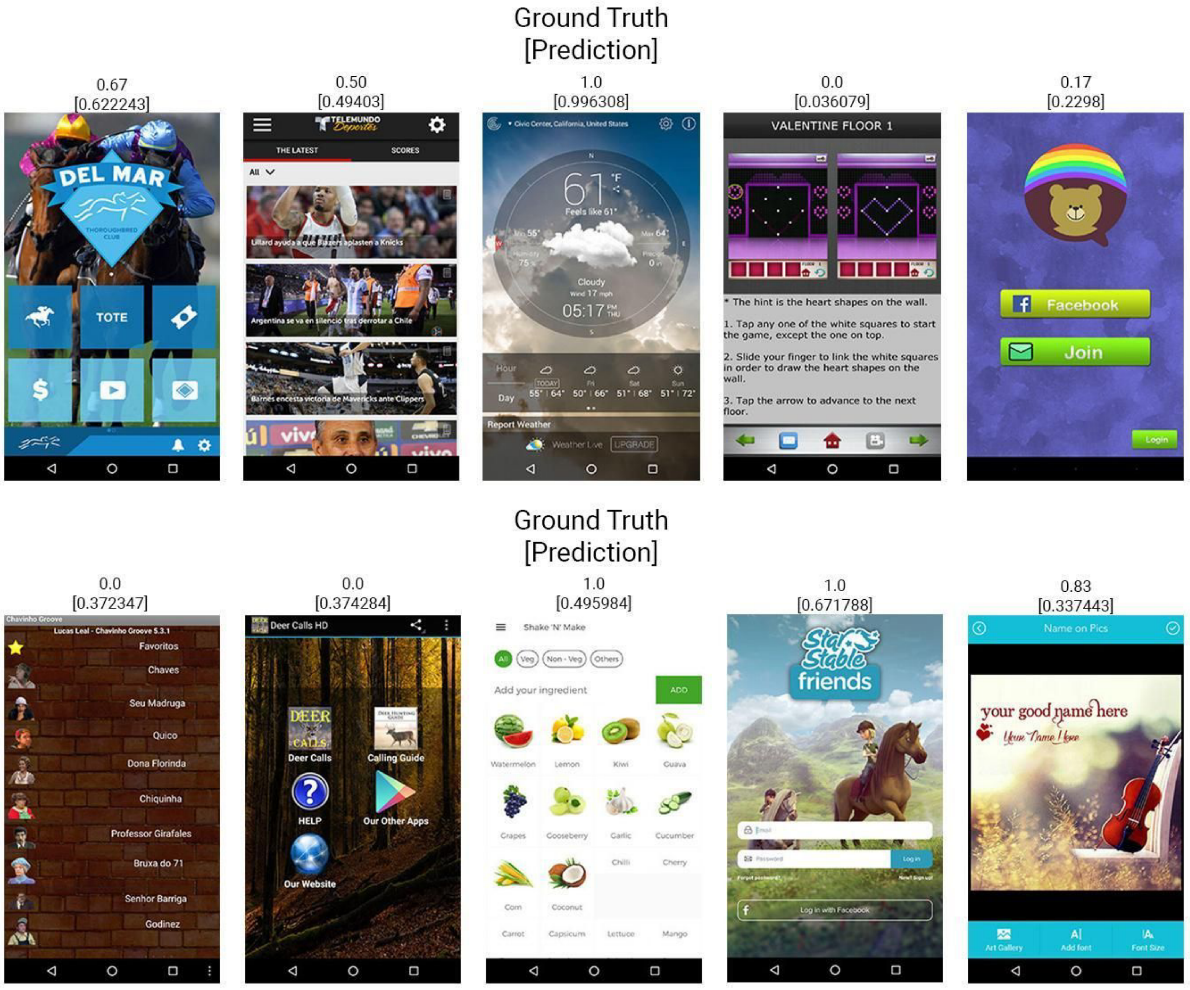

Fig. 5. Comparison of human ratings and *Appsthetics* predictions (top row: high correspondence; bottom row: low correspondence).

The *Appthetics* model seems to assign higher weights to colors and variations for prediction, as seen on the bottom row of Fig. 5. Even though humans rated the visual aesthetics of the first two screenshots very low, the model predicts a fairly higher value most likely due to its colors. Likewise, the last two screenshots on the bottom row were rated highly by the humans but low by the model, most likely due to their color pallet. Fig. 6 further supports this hypothesis by displaying how the model predicts low scores to screenshots with mostly white content and considerably higher scores to images with mostly black or small variations of the same tone. Yet, on the other hand, the examples also illustrate that visual aesthetics are not influenced only by color, as some MUIs using few colors are still rated low.

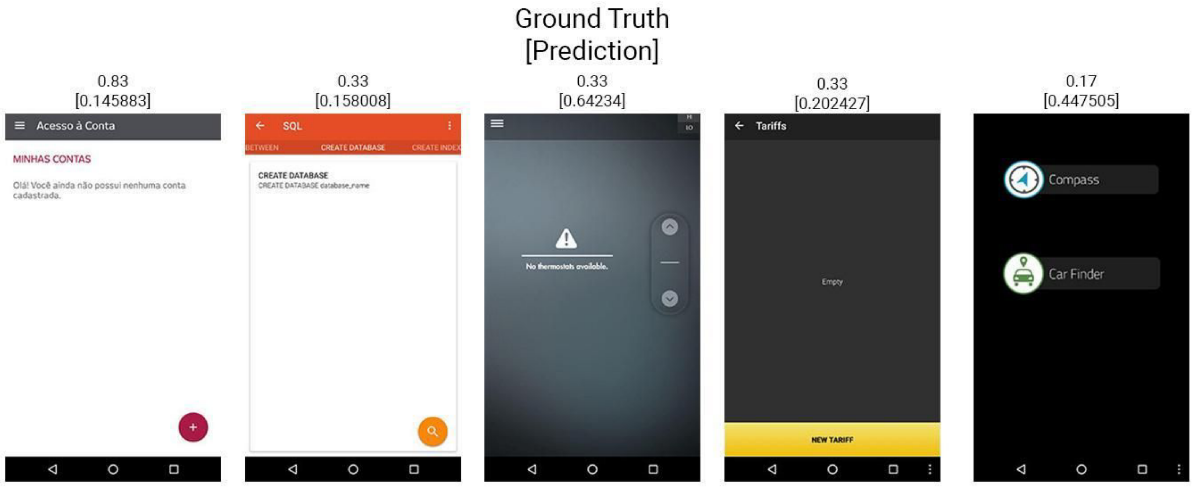

Fig. 6. Low variation in colors used.

These examples also demonstrate the reported difficulty of identifying the factors that influence the perception of visual aesthetics, such as simplicity, as, e.g., some MUIs with very sparse content are highly rated whereas others are considered of low aesthetics. This also further indicates the potential of deep learning approaches for this complex task as they are able to implicitly compose multiple factors and make predictions with acceptable accuracy. The obtained results show that deep learning can be applied successfully for this task, representing a step towards the automation of the aesthetics assessment of MUIs and thus contributing to the improvement of their quality and the development process. We are currently working on the deployment of the *Appsthetics* model, as well as it's continuous improvement through the evolution of the dataset.

Threats to validity. A potential threat to the results of our study is related to the dataset used. In order to minimize this threat, we tried to balance the dataset with respect to the aesthetic ratings. Test and evaluation have been performed on the test set (not previously used for learning) that has been randomly chosen from the dataset. A further threat comes from the variations of interrater agreements during labeling. In order to minimize this threat, we only used MUIs with acceptable interrater agreement for learning and testing. For evaluation we selected appropriate methods in accordance with related work as well as theory in order to evaluate correlation and agreement. With respect to external validity, we used a considerable sample size for evaluation with a large variety of types of applications in order to allow the generalization of the results.

7. Conclusion

In this article, we present an innovative approach for automatically assessing the visual aesthetics of user interfaces of Android apps adopting deep learning techniques. In order to train the model, we created a dataset with 3,139 Android user interfaces associated with a human rating on their visual aesthetics. We have demonstrated that the *Appsthetics* model is able to effectively provide aesthetics predictions with high correlations and agreement with the human ratings. The proposed model can be used for the effective and efficient assessment of aesthetics during the user interface design process during the development of Android apps. It also showcases a further example of how deep learning technology can be applied in order to advance software engineering and human-computer interaction. We now intend to deploy the model into an automatic assessment tool and to improve its performance based on continuous feedback.

Acknowledgment

We would like to thank all participants for their help to rate the aesthetics of the MUIs.

Funding: This work was supported by CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico – www.cnpq.br), an entity of the Brazilian government focused on scientific and technological development.

8. References

Alemerien, K. and Magel, K. (2014). GUIEvaluator: A Metric-tool for Evaluating the Complexity of Graphical User Interfaces. In Proceedings of the Twenty-Sixth International Conference on Software Engineering & Knowledge Engineering, pp. 13-18. Knowledge Systems Institute Graduate School.

Alsudani, F. and Casey, M. (2009). The effect of aesthetics on web credibility. In Proceedings of the 23rd British HCI Group annual conference on people and computers: Celebrating people and technology, pp. 512-519. British Computer Society.

Amershi, S., Begel, A., Bird, C., DeLine, R., Gall, H., Kamar, E., Nagappan, N., Nushi, B. and Zimmermann, T. (2019). Software engineering for machine learning: a case study. Proceedings of the 41st International Conference on Software Engineering: Software Engineering in Practice, pp. 291-300. IEEE Press.

Anderson, S. P. (2011). Seductive interaction design: Creating playful, fun, and effective user experiences. Pearson Education, San Francisco.

Ashmore, R., Calinescu, R. and Paterson, C. (2019). Assuring the Machine Learning Lifecycle: Desiderata, Methods, and Challenges. arXiv preprint arXiv:1905.04223.

Balagtas-Fernandez, F., Forrai, J. and Hussmann, H., (2009). Evaluation of user interface design and input methods for applications on mobile touch screen devices. In Proceedings of the IFIP Conference on Human-Computer Interaction, pp. 243-246. Springer, Berlin.

Ben-Bassat, T., Meyer, J. and Tractinsky, N. (2006). Economic and subjective measures of the perceived value of aesthetics and usability. ACM Transactions on Computer-Human Interaction, 13(2), 210-234.

Bhandari, U., Neben, T., Chang, K. and Chua, W. Y. (2017). Effects of interface design factors on affective responses and quality evaluations in mobile applications. Computers in Human Behavior, 72, 525-534.

Bhandari, U., Chang, K. and Neben, T. (2019). Understanding the impact of perceived visual aesthetics on user evaluations: An emotional perspective. Information & Management, 56(1), 85-93.

Bland, J. M. and Altman, D. (1986). Statistical methods for assessing agreement between two methods of clinical measurement. The Lancet, 327(8476), 307-310.

Bloch, P. H., 1995. Seeking the ideal form: Product design and consumer response. Journal of Marketing, 59(3), 16-29.

Bonett, D. G. and Wright, T. A.,(2000). Sample size requirements for estimating Pearson, Kendall and Spearman correlations. Psychometrika, 65(1), 23-28.

Botzenhardt, A., Li, Y. and Maedche, A. (2016). The roles of form and function in utilitarian mobile data service design. Journal of Electronic Commerce Research, 17(3), 220.

Chang, H. H. and Chen, S. W. (2009). Consumer perception of interface quality, security, and loyalty in electronic commerce. Information & management, 46(7), 411-417.

Choi, J. H. and Lee, H. J. (2012). Facets of simplicity for the smartphone interface: A structural model. International Journal of Human-Computer Studies, 70(2), 129-142.

Coursaris, C. K. and van Osch, W. (2016). A Cognitive-Affective Model of Perceived User Satisfaction (CAMPUS): The complementary effects and interdependence of usability and aesthetics in IS design. Information & Management, 53(2), 252-264.

Deng, Y., Loy, C. C. and Tang, X. (2017). Image aesthetic assessment: An experimental survey. IEEE Signal Processing Magazine, 34(4), 80-106.

Dou, Q., Zheng, X. S., Sun, T. and Heng, P. A. (2019). Webthetics: quantifying webpage aesthetics with deep learning. International Journal of Human-Computer Studies, 124, 56-66.

Fast.ai, (2017). https://github.com/fastai/fastai. (accessed 6 November 2019).

Fonnegra, R. D., Blair, B. and Díaz, G. M. (2017). Performance comparison of deep learning frameworks in image classification problems using convolutional and recurrent networks. In Proceedings of IEEE Colombian Conference on Communications and Computing, pp. 1-6. IEEE.

Giavarina, D. (2015). Understanding bland altman analysis. Biochemia medica: Biochemia medica, 25(2), 141-151.

Gupta, V. S. and Kohli, S. (2015). Automated Interestingness Calculator for mobile app recommendation. In Proceedings of the 4th International Conference on Reliability, Infocom Technologies and Optimization, pp. 1-6. IEEE.

Hamborg, K. C., Hülsmann, J. and Kaspar, K. (2014). The interplay between usability and aesthetics: More evidence for the "what is usable is beautiful" notion. Advances in Human-Computer Interaction, 15.

Hartmann, J., Sutcliffe, A. and De Angeli, A. (2007). Investigating attractiveness in web user interfaces. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 387-396. ACM.

He, K., Zhang, X., Ren, S. and Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770-778, IEEE.

Howard, J., (2018). fastai v1 for PyTorch: Fast and accurate neural nets using modern best practices. https://www.fast.ai/2018/10/02/fastai-ai/. (accessed 6 November 2019).

Huang, K. Y., (2009). Challenges in human-computer interaction design for mobile devices. In Proceedings of the World Congress on Engineering and Computer Science, pp. 236-241, Newswood Limited.

IDC, (2019). Smartphone Market Share. https://www.idc.com/promo/smartphone-market-share/os. (accessed 6 November 2019).

ISO/IEC 25010:2011. Systems and software engineering -- Systems and Software Quality Requirements and Evaluation (SQuaRE) -- System and software quality models.

Khani, M. G., Mazinani, M. R., Fayyaz, M. and Hoseini, M. (2016). A novel approach for website aesthetic evaluation based on convolutional neural networks. In Proceedings of the Second International Conference on Web Research, pp. 48-53. IEEE.

Kim, J., Lee, J. and Choi, D. (2003). Designing emotionally evocative homepages: an empirical study of the quantitative relations between design factors and emotional dimensions. International Journal of Human-Computer Studies, 59(6), 899-940.

Kumar, D. S., Purani, K. and Viswanathan, S. A. (2018). Influences of 'appscape' on mobile app adoption and m-loyalty. Journal of Retailing and Consumer Services, 45, 132-141.

LeCun, Y., Bengio, Y. and Hinton, G. (2015). Deep learning. Nature, 521(7553), p. 436-444.

Lindgaard, G. and Dudek, C. (2002). User satisfaction, aesthetics and usability. In Proceedings of the IFIP World Computer Congress, TC 13, pp. 231-246. Springer, Boston, MA.

Lindgaard, G., Fernandes, G., Dudek, C. and Brown, J. (2006). Attention web designers: You have 50 milliseconds to make a good first impression!. Behaviour & Information Technology, 25(2), 115-126.

Lindgaard, G., Dudek, C., Sen, D., Sumegi, L. and Noonan, P. (2011). An exploration of relations between visual appeal, trustworthiness and perceived usability of homepages. ACM Transactions on Computer-Human Interaction, 18(1), 1.

Litchfield, S. (2010), Defining the Smartphone - part 1. http://www.allaboutsymbian.com/features/item/Defining_the_Smartphone.php. (access 6 November 2019).

Lu, Y., Tan, B. and Wang, Y. (2013). Web aesthetics: How does it influence the sales performance in online marketplaces.

Lu, X., Lin, Z., Jin, H., Yang, J. and Wang, J. Z. (2014). Rapid: Rating pictorial aesthetics using deep learning. In Proceedings of the 22nd ACM International Conference on Multimedia, pp. 457-466. ACM.

Lu, X., Lin, Z., Jin, H., Yang, J. and Wang, J. Z. (2015). Rating image aesthetics using deep learning. IEEE Transactions on Multimedia, 17(11), 2021-2034.

Malu, G., Bapi, R. S. and Indurkhya, B. (2017). Learning photography aesthetics with deep cnns. arXiv preprint arXiv:1707.03981.

Martinez, J., Sottet, J. S., Frey, A. G., Ziadi, T., Bissyandé, T., Vanderdonckt, J., Klein, J. and Le Traon, Y. (2017). Variability management and assessment for user interface design. In Human-Centered Software Product Lines, pp. 81-106. Springer, Cham.

Miniukovich, A. and De Angeli, A. (2014a). Quantification of interface visual complexity. In Proceedings of the International Working Conference on Advanced Visual Interfaces, pp. 153-160. ACM.

Miniukovich, A. and De Angeli, A. (2014b). Visual impressions of mobile app interfaces. In Proceedings of the 8th Nordic Conference on Human-Computer Interaction: Fun, Fast, Foundational, pp. 31-40. ACM.

Miniukovich, A. and De Angeli, A. (2015). Computation of interface aesthetics. In Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems, pp. 1163-1172. ACM.

Mitchell, T. M. (1997). Machine Learning. First ed. McGraw-Hill Education, New York.

Moshagen, M. and Thielsch, M. T. (2010). Facets of visual aesthetics. International Journal of Human-Computer Studies, 68(10), 689-709.

Myers, B. A. and Rosson, M. B. (1992). Survey on user interface programming. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 195-202. ACM.

Norman, D. A. (2002). Emotion Design: Attractive Things Work Better. Interactions Magazine. 9. 36-42.

Oyibo, K., Orji, R. and Vassileva, J. (2017). The influence of personality on mobile web credibility. In Adjunct Publication of the 25th Conference on User Modeling, Adaptation and Personalization, pp. 53-58. ACM.

Polyzotis, N., Roy, S., Whang, S. E. and Zinkevich, M. (2018). Data lifecycle challenges in production machine learning: a survey. ACM SIGMOD Record, 47(2), 17-28.

Reinecke, K., Yeh, T., Miratrix, L., Mardiko, R., Zhao, Y., Liu, J. and Gajos, K. Z. (2013). Predicting users' first impressions of website aesthetics with a quantification of perceived visual complexity and colorfulness. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 2049-2058. ACM.

Ripley, B. D. and Hjort, N. L. (1996). Pattern Recognition and Neural Networks. Cambridge University Press, New York, USA.

Roast, C., Evriviades, M., Purcell, M. and Steele, B. (2002). Interaction media-using IT and liking IT. In Proceedings of the Pan Hellenic Conference on Human-Computer Interaction, ACM.

Robins, D. and Holmes, J. (2008). Aesthetics and credibility in web site design. Information Processing & Management, 44(1), 386-399.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M. and Berg, A. C. (2015). Imagenet large scale visual recognition challenge. International Journal of Computer Vision, 115(3), 211-252.

Schenkman, B. N. and Jönsson, F. U. (2000). Aesthetics and preferences of web pages. Behaviour & Information Technology, 19(5), 367-377.

Seckler, M., Opwis, K. and Tuch, A. N. (2015). Linking objective design factors with subjective aesthetics: An experimental study on how structure and color of websites affect the facets of users' visual aesthetic perception. Computers in Human Behavior, 49, 375-389.

Silvennoinen, J., Vogel, M. and Kujala, S. (2014). Experiencing visual usability and aesthetics in two mobile application contexts. Journal of Usability Studies, 10(1), 46-62.

Smith, L. N. (2018). A disciplined approach to neural network hyper-parameters: Part 1--learning rate, batch size, momentum, and weight decay. arXiv preprint arXiv:1803.09820.

Smith, L. N. and Topin, N. (2018). Super-convergence: Very fast training of residual networks using large learning rates.

Sonderegger, A. and Sauer, J., (2010). The influence of design aesthetics in usability testing: Effects on user performance and perceived usability. Applied Ergonomics, 41(3), 403-410.

Statcounter (2019). Mobile Operating System Market Share Worldwide. https://gs.statcounter.com/os-market-share/mobile/worldwide. (accessed 6 November 2019).

Suchecki, M. and Trzciski, T. (2017). Understanding aesthetics in photography using deep convolutional neural networks. In Proceedings of the Signal Processing: Algorithms, Architectures, Arrangements, and Applications, pp. 149-153. IEEE.

Taba, S. E. S., Keivanloo, I., Zou, Y., Ng, J. and Ng, T. (2014). An exploratory study on the relation between user interface complexity and the perceived quality. In Proceedings of the International Conference on Web Engineering, pp. 370-379. Springer, Cham.

Tractinsky, N., Katz, A. S. and Ikar, D. (2000). What is beautiful is usable. Interacting with computers, 13(2), 127-145.

Tractinsky, N., Cokhavi, A., Kirschenbaum, M. and Sharfi, T. (2006). Evaluating the consistency of immediate aesthetic perceptions of web pages. International Journal of Human-Computer Studies, 64(11), 1071-1083.

Tractinsky, N. (2013). The Encyclopedia of Human Interaction, 2nd ed.: Visual Aesthetics. https://www.interaction-design.org/literature/book/the-encyclopedia-of-human-computer-interaction-2nd-ed/visual-aesthetics (accessed 6 November 2019).

Tuch, A. N., Roth, S. P., HornbæK, K., Opwis, K. and Bargas-Avila, J. A. (2012). Is beautiful really usable? Toward understanding the relation between usability, aesthetics, and affect in HCI. Computers in Human Behavior, 28(5), 1596-1607.

Wasserman, T. (2010). Software engineering issues for mobile application development. In Proceedings of the Workshop on Future of Software Engineering Research, Santa Fe, USA. ACM.

Xu, C., Peak, D. and Prybutok, V. (2015). A customer value, satisfaction, and loyalty perspective of mobile application recommendations. Decision Support Systems, 79, 171-183.

Yosinski, J., Clune, J., Bengio, Y. and Lipson, H. (2014). How transferable are features in deep neural networks?. In Proceedings of the Advances in Neural Information Processing Systems, pp. 3320-3328. Curran Associates, Inc.

Zen, M. and Vanderdonckt, J. (2016). Assessing user interface aesthetics based on the inter-subjectivity of judgment. In Proceedings of the 30th International BCS Human-Computer Interaction Conference: Fusion!, p. 25. BCS Learning & Development Ltd.

Zheng, X. S., Chakraborty, I., Lin, J. J. W. and Rauschenberger, R. (2009). Correlating low-level image statistics with users-rapid aesthetic and affective judgments of web pages. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 1-10. ACM.