UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
CIÊNCIAS DA COMPUTAÇÃO

LEONARDO KREUCH

Desenvolvimento de um Modelo de Avaliação da Originalidade do Esqueleto de Design de Interface de Aplicativos Android

FLORIANÓPOLIS
2022

LEONARDO KREUCH

Desenvolvimento de um Modelo de Avaliação da Originalidade do Esqueleto de Design de Interface de Aplicativos Android

Trabalho de Conclusão do Curso de Graduação em Ciências da Computação do Centro de Tecnológico da Universidade Federal de Santa Catarina como requisito para a obtenção do título de Bacharel em Ciências da Computação.
Orientadora: Prof.ª Dr.ª rer. nat. Christiane Gresse von Wangenheim, PMP
Coorientadora: Prof.ª Nathalia da Cruz Alves

FLORIANÓPOLIS
2022

RESUMO

Com o avanço da tecnologia no século XXI, novas habilidades se fazem necessárias na vida cotidiana e no mercado de trabalho. Dentre essas habilidades, destaca-se a criatividade, que é a capacidade de criar algo novo e útil. Uma das formas de estimular a criatividade é por meio da computação na Educação Básica, utilizando ambientes de programação em blocos como o *App Inventor*. Além da programação em si, outro aspecto importante é o design de interface de aplicativos para o usuário, que afeta diretamente a qualidade de uso do aplicativo pelo usuário. Por isso, como parte do processo de ensino-aprendizagem torna-se importante também, a avaliação da estética visual dos aplicativos criados pelos alunos. Nesse contexto, o objetivo deste trabalho é o desenvolvimento de um modelo automatizado, utilizando técnicas de *deep learning,* para avaliar a originalidade no design de interface de aplicativos desenvolvidos com o ambiente *App Inventor*, para ser utilizado no ensino de computação na Educação Básica. O modelo desenvolvido possibilita a avaliação da originalidade dos apps desenvolvidos como resultados do processo de aprendizagem de forma automatizada, assim, espera-se contribuir para melhorar a habilidade de criatividade, fornecendo feedback aos alunos e professores no processo de aprendizagem.

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

ABSTRACT

With the advance of technology in the 21st century, new skills are required in everyday life and in the labor market. Among these skills is creativity, which is the ability to create something new and useful. One of the ways to stimulate creativity is through computing in Basic Education, using block programming environments such as App Inventor. Besides the programming itself, another important aspect is the user interface design of applications, which directly affects how well the application is used by the user. Therefore, as part of the teaching-learning process it is also important to evaluate the visual aesthetics of the applications created by the students. In this context, the objective of this work is to develop an automated model, using deep learning techniques, to evaluate the originality in interface design of applications developed with the App Inventor environment, to be used in teaching computer science in Basic Education. The developed enables the evaluation of the originality of the applications developed as a result of the learning process in a designed way, thus, it is expected to contribute to improve a creativity skill, in the learning process of learning to the students and model.

Keywords: Originality, Creativity, Visual Aesthetics, App Inventor, User Interface Design, Basic Education, Deep Learning.

LISTA DE FIGURAS

Figura 1 - 4Ps: vertentes da criatividade 15
Figura 2 - Abordagem UMC no contexto de desenvolvimento de apps 17
Figura 3 - Rubrica Modifique e Crie com o Universo de Comparação do Produto 19
Figura 4 - Tela “Blocos” do App Inventor.....21
Figura 5 - Tela “Designer” do App Inventor 22
Figura 6 - Diferentes alinhamentos à esquerda para a mesma tela 24
Figura 7 - Diferentes alinhamentos no centro para a mesma tela 24
Figura 8 - Diferentes alinhamentos à direita para a mesma tela 25
Figura 9 - Relação entre Deep Learning e a área de inteligência artificial 27
Figura 10 - Relação de sistemas de IA em diferentes disciplinas 28
Figura 11 - Diagrama exemplo de um nó 29
Figura 12 - Estrutura básica de uma rede neural 30
Figura 13 - Hierarquia de características, com aumento da complexidade e abstração a cada nível 31
Figura 14 – Detecção de objetos.....33
Figura 15 - Exemplo avaliação 45
Figura 16 - Fluxo de trabalho idealizado 49
Figura 17 - Detecção de objetos 50
Figura 18 - Frequência do uso dos componentes de interface de usuário no App Inventor 51
Figura 19 - Visualização dos componentes selecionados no App Inventor 52
Figura 20 - Exemplos de *screenshots* selecionados 53
Figura 21 - Exemplo de *screenshot* e arquivo com o conjunto de labels 54
Figura 22 - Frequência dos componentes de interface no conjunto de dados.....54
Figura 23 - Métricas de desempenho do treinamento da rede 56
Figura 24 - Visão geral das medidas de desempenho 57
Figura 25 - Matriz de confusão do modelo treinado 58
Figura 26 - Grau de predição que a rede classifica cada elemento de uma tela.....59
Figura 27 - Exemplo de tela poluída.....62
Figura 28 - Exemplo de tela com classificação errônea.....63
Figura 29 - Curva sobre as notas de originalidade alocada por humanos 70

LISTA DE QUADROS

Quadro 1 - Componentes de design de interface de usuário disponíveis no App Inventor 22
Quadro 2 - Termos de busca relevantes, sinônimos e tradução (inglês).....35
Quadro 3 - Strings de busca utilizadas nas diferentes bases de dados 36
Quadro 4 - Resultados da busca.....37
Quadro 5 - Documentos resultantes da execução de busca. 38
Quadro 6 - Tipo dos critérios de design interface em cada abordagem 39
Quadro 7 - Dados extraídos para responder à Pergunta de Análise 4.....40
Quadro 8 - Dados extraídos para responder à Pergunta de Análise 5.....433
Quadro 9 - Componentes considerados na avaliação 51
Quadro 10 -Parâmetros utilizados no treinamento do modelo 55
Quadro 11 - Frequência de ocorrência de componentes no universo de referência .64
Quadro 12 - Comparação dos rankings de similaridade 67
Quadro 13 - Coeficientes de correlação de Spearman 68
Quadro 14 - Valores de similaridade e notas de originalidade para o conjunto de testes.....69

SUMÁRIO

1 INTRODUÇÃO 9

1.1 CONTEXTUALIZAÇÃO 9

1.2 OBJETIVOS 11

1.2.1 Objetivo Geral 11

1.2.2 Objetivos Específicos 11

1.3 METODOLOGIA DE PESQUISA 12

1.4 ESTRUTURA DO DOCUMENTO 13

2 FUNDAMENTAÇÃO TEÓRICA 14

2.1 ORIGINALIDADE NA CRIATIVIDADE 14

2.1.1 Criatividade do produto 15

2.1.2 Originalidade no contexto da criatividade no ensino da computação 16

2.2 DESIGN VISUAL DE APLICATIVOS MÓVEIS 19

2.3 ORIGINALIDADE DO DESIGN DE INTERFACE DE APPS 20

2.4 DEEP LEARNING 25

2.4.1 Redes Neurais 28

2.4.2 *Deep Neural Network (DNN)* 30

2.4.3 *Yolo* 32

3 ESTADO DA ARTE 34

3.1 DEFINIÇÃO DO PROTOCOLO DE MAPEAMENTO 34

3.2 EXECUÇÃO DE BUSCA 36

3.3 ANÁLISE DE RESULTADOS 37

3.3.1 Quais modelos/ferramentas de avaliação da originalidade do design de interfaces de apps Android existem? 37

3.3.2 Quais as características do modelo de avaliação de originalidade do design de interfaces? 39

3.3.3 Como foi feita a preparação do conjunto de dados e a rotulação? ... 40

3.3.4 Como o modelo/ferramenta foi avaliado? 42
3.3.5 Como a qualidade do modelo foi avaliada? 43
3.4 DISCUSSÃO 46
3.4.1 Ameaças à validade 47
4 MODELO DE AVALIAÇÃO DA ORIGINALIDADE DE DESIGN DE INTERFACE DE APLICATIVOS ANDROID 48
4.1 ANÁLISE DE REQUISITOS 48
4.2 CRIAÇÃO DE UM MODELO DE DETECÇÃO DE ELEMENTOS E LAYOUT DE UI 49
4.2.1 Detecção e seleção de componentes 49
4.2.2 Preparação do conjunto de dados 52
4.2.3 Treinamento do modelo de detecção de componentes 55
4.2.4 Avaliação do desempenho do modelo de detecção de objetos 56
4.3 ANÁLISE DO GRAU DE SIMILARIDADE 63
4.3.1 Análise do grau de similaridade de uma tela 63
4.3.1.1 Criação do universo de referência 64
4.3.1.2 Cálculo do valor de similaridade 64
4.3.2 Avaliação do desempenho do cálculo do valor de similaridade de um app 66
4.4 CÁLCULO DA NOTA DE ORIGINALIDADE 68
4.5 IMPLEMENTAÇÃO DA SOLUÇÃO 70
5 CONCLUSÃO 72
REFERÊNCIAS 73
APÊNDICE A 82
APÊNDICE B 83

1 INTRODUÇÃO

1.1 CONTEXTUALIZAÇÃO

A criatividade é considerada uma das principais competências do século XXI, essencial para o sucesso profissional e pessoal (P21, 2019). Segundo Runco e Jaeger (2012) esta competência tem uma definição padrão dividida em 2 características que são ressaltadas necessárias para **criatividade** como a originalidade e a eficácia. A criatividade pode se referir a quatro diferentes pontos de vista: pessoas, processos, imprensa e produtos (RHODES, 1961). A vertente produtos é a que será abordada neste trabalho. Ela se refere aos resultados tangíveis do processo criativo (ou seja, obras de arte, programas de computador entre outras). Assim, um produto criativo pode ser considerado algo que é original, útil e eficaz (RUNCO; JAEGER, 2012), sendo a originalidade uma das principais características da criatividade. Uma ideia ou produto criativo é considerado original se representar algo novo ou que não existia antes (RUNCO; JAEGER, 2012).

Como a criatividade faz parte das habilidades do século XXI, promover seu ensino desde a Educação Básica é muito importante (P21, 2019). Geralmente, a criatividade está associada às artes e músicas, porém, ela pode ser ensinada também por meio da computação (ALVES; GRESSE VON WANGENHEIM; MARTINS-PACHECO; BORGATTO, 2022). A computação tem o potencial para propiciar aos usuários oportunidades para utilizar suas expressões criativas para resolver problemas, criar artefatos computacionais, e desenvolver novos conhecimentos (YADAV; COOPER, 2017).

A computação na Educação Básica é tipicamente ensinada com enfoque no ensino de algoritmos e programação usando metodologias ativas, levando os alunos a criar artefatos computacionais como jogos e apps por meio de ambientes de programação visual baseados em blocos, como o *Scratch* ou *App Inventor*. Uma das alternativas consiste em ensinar o desenvolvimento de apps com o *App Inventor* (MUSTAFARAJ; TURBAK; SVANBERG, 2017), que suporta tanto a programação da parte funcional quanto o design de interface de usuário, para a criação de apps Android. Assim, os estudantes aprendem tanto conceitos de algoritmos e programação como também de design de interface, que fazem parte do corpo de

conhecimento de computação (ACM; IEEE; IEEE Computer Society, 2013). Já que o design da interface visa maximizar a usabilidade e a experiência do usuário (COOPER, 2014) e afeta diretamente a qualidade de uso do aplicativo pelo usuário (NIELSEN, 1994).

Para ensinar o desenvolvimento de apps na Educação Básica é tipicamente adotado o ciclo "*Use-Modify-Create*" (LEE *et al*., 2011) guiando a progressão de aprendizagem. No estágio *Use*, os alunos seguem um tutorial criando um artefato computacional predefinido. No estágio *Modify*, os alunos modificam esse artefato, alterando características e criando novas funcionalidades. Progressivamente, os alunos ganham confiança e avançam para o estágio *Create*, no qual eles são incentivados a desenvolverem seus próprios artefatos computacionais com as características que lhes convém, abordando temas de sua escolha. Assim, os alunos desenvolvem a capacidade de gerar ideias e soluções originais e úteis durante as etapas de modificação e principalmente na criação (WALIA, 2019).

Para acompanhar o desenvolvimento da criatividade dos alunos, é fundamental haver uma forma de fornecer um *feedback* (MISHRA; HENRIKSEN, 2013), incluindo também um *feedback* sobre a originalidade do design de interface de usuário criada pelo aluno como resultado da aprendizagem. Entretanto, o processo de avaliação da criatividade é um desafio, pois a criatividade é difícil de medir de forma objetiva com confiabilidade e validade (HENRIKSEN; MISHRA; MEHTA, 2015).

Esse processo de avaliação requer um esforço considerável por parte do avaliador e do risco de introduzir um viés se for feito por meio de um julgamento humano, podendo ser influenciado por aspectos pessoais e valores culturais (GRESSE VON WANGENHEIM et al., 2018b; ALVES *et. al,* 2021). Assim, uma alternativa é automatizar o processo de avaliação da originalidade do design de interface para aumentar a confiabilidade e validade dos resultados da avaliação, além de minimizar o tempo e esforço dos professores da Educação Básica referente a essa tarefa.

Portanto, o objetivo dessa pesquisa é definir um modelo de avaliação automatizado da originalidade do produto desenvolvido por alunos usando *App Inventor* adotando técnicas de *deep learning*, a fim de avaliar a originalidade da interface de um aplicativo.

Com relação às premissas e restrições, o trabalho foi 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 originalidade de interfaces não abordando outras qualidades. O trabalho também foca na análise de aplicativos Android não abordando interfaces de apps de outras plataformas. A automatização enfocará em projetos desenvolvidos com a ferramenta de desenvolvimento App Inventor no ensino de computação na educação básica (especificamente nos anos finais do ensino fundamental).

1.2 OBJETIVOS

Nas seções abaixo estão descritos o objetivo geral e os objetivos específicos deste Trabalho de Conclusão de Curso.

1.2.1 Objetivo geral

O objetivo geral deste trabalho é desenvolver um modelo de análise da originalidade de esqueleto de design de interface de apps Android desenvolvidos com App Inventor. Como base nos *screenshots* das telas dos apps são automaticamente detectados os elementos de UI e a sua posição. São adotadas técnicas para comparar o esqueleto do design de interface de novo app com os esqueletos de design de interface de apps de um universo de referência e identificado um grau de similaridade/diferença. E, a partir desse grau de similaridade/diferença é calculada uma nota de originalidade dentro do contexto da Educação Básica (anos finais do Ensino Fundamental).

1.2.2 Objetivos Específicos

O1. Analisar a fundamentação teórica sobre originalidade, design de interfaces de usuário e *deep learning*.

O2. Analisar o estado da arte em relação a análise automática da originalidade de design de design de interfaces de apps.

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

1.3 METODOLOGIA DE PESQUISA

A metodologia de pesquisa utilizada neste trabalho é dividida nas seguintes etapas.

Etapa 1 – Fundamentação teórica

Estudando, analisando e sintetizando os conceitos principais e a teoria referente aos temas a serem abordados neste trabalho é apresentada a fundamentação teórica utilizando a metodologia de revisão narrativa (CORDEIRO *et al.*, 2007). Nesta etapa são realizadas as seguintes atividades:

A1.1 – Análise teórica sobre originalidade no contexto da criatividade

A1.2 - Análise teórica sobre design de interface de apps

A1.3 – 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, Vakkalanka e Kuzniarz (2015) para identificar e analisar modelos de análise automatizada da originalidade de design de interfaces de usuário de apps atualmente sendo utilizados. Esta etapa é dividida nas seguintes atividades:

A2.1 – Definição do protocolo da revisão

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 originalidade de design de interfaces de apps criados no App Inventor, seguindo um processo de desenvolvimento de redes neurais/*deep learning* (KIERSKI, 2017; POLYZOTIS *et al*., 2017). Esta etapa é dividida nas seguintes atividades:

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 – Ajuste fino de parâmetros
A3.8 – Predição/Inferência

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 são apresentados o estado da arte, a situação atual em que se encontram os trabalhos e propostas existentes na área de avaliação de similaridade do design visual de interfaces de apps móveis. O capítulo 4 apresenta o desenvolvimento do modelo, começando pela análise de requisitos até a avaliação de desempenho do modelo. O capítulo 5 apresenta a conclusão do presente trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são apresentados conceitos essenciais e relevantes ao trabalho desenvolvido. Iniciando por uma análise teórica sobre a originalidade na criatividade, seguido por uma análise teórica sobre design visual e, por fim, sobre inteligência artificial, com foco em *deep learning.*

2.1 ORIGINALIDADE NA CRIATIVIDADE

A criatividade é considerada uma das principais competências do século XXI, essencial para o sucesso profissional e pessoal (P21, 2015; KAUFMAN; BEGHETTO, 2009). A criatividade vem se tornando uma das habilidades centrais nos currículos de vários países ao redor do mundo (BEGHETTO, 2010) e muito mais deve ser feito para criar oportunidades de expressão criativa nos currículos (KAUFMAN; BEGHETTO, 2013). Promover seu ensino desde a Educação Básica é muito importante (P21, 2020; KAUFMAN; BEGHETTO, 2013) embora existam diversas dificuldades de avaliar a criatividade e de incentivá-las por parte dos professores (KAUFMAN; BEGHETTO, 2013).

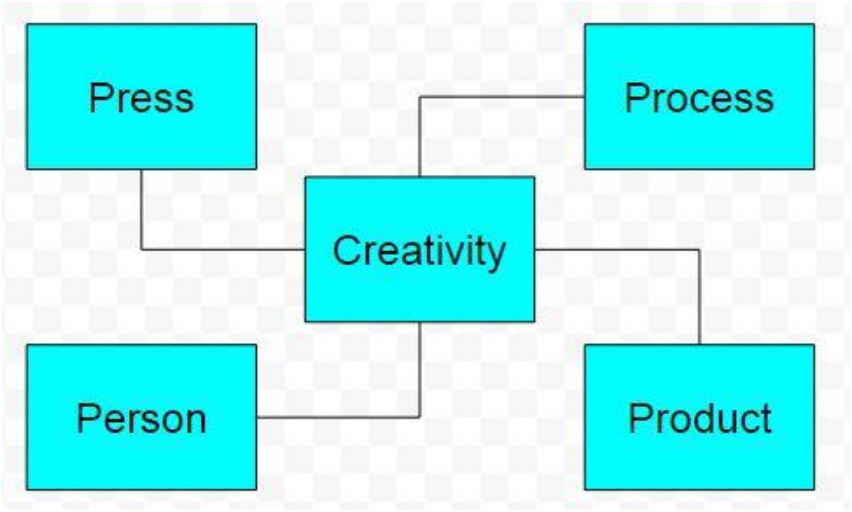
Entretanto, ter uma definição sobre a criatividade não é algo trivial, pois existem divergências que levam à falta de um padrão e a um entendimento inconsistente (WALIA, 2019). Características como originalidade, novidade e eficácia são ressaltadas como necessárias para a criatividade (RUNCO; JAEGER, 2012; MISHRA; HENRIKSEN, 2013). Porém, as características isoladamente não tornam uma ideia criativa. Uma ideia original inútil não é suficiente para a criatividade, ideias originais tem que ser eficazes e ter utilidade para serem criativas (RUNCO; JAEGER, 2012).

Buscando uma melhor forma de estruturar a criatividade, Rhodes (1961) separou a criatividade em 4 vertentes (Figura 1):

- Criatividade de Pessoa refere-se ao indivíduo, onde suas características como personalidade, hábitos e atitudes influenciam no seu potencial criativo.
- Criatividade de Processo refere-se ao esforço mental durante aprendizagem, comunicação e pensamento criativo.

- Criatividade de Ambiente refere-se ao ambiente, ideias criativas em resposta às necessidades e percepções que o indivíduo recebe de fontes internas e externas.
- Criatividade de Produto refere-se aos resultados tangíveis do processo criativo (ou seja, obras de arte, livros, programas de computador, etc).

Figura 1 - 4Ps: vertentes da criatividade



Fonte: RHODES, 1961

Conforme o foco do presente trabalho, está sendo detalhada na próxima seção a criatividade do produto.

2.1.1 Criatividade do produto

Uma ideia ou produto criativo foi o pensamento do homem em um determinado momento do tempo (RHODES, 1961). A avaliação da criatividade pode ser feita por meio de uma análise das propriedades do produto criativo (JACKSON; MESSICK, 1964). A análise do produto criativo permite rastrear os pensamentos e os eventos que levaram o homem àquela ideia naquele determinado momento. Embora a criatividade não possa ser classificada apenas pelo ponto de vista do produto (RITCHIE, 2001), avaliar a criatividade pela abordagem do produto oferece formas de

criar referências desse conceito por meio da sistematização de produções das pessoas. (SKAGER; SCHULTZ; KLEIN, 1966).

Normalmente as abordagens para avaliar um produto criativo sugerem algumas características. Entretanto, não há consenso sobre quais características fazem parte do construto abstrato de criatividade de produto, embora os fatores mais confiáveis tenham sido vastamente discutidos (RANJAN; SIDDHARTH.; CHAKRABARTI, 2018). Originalidade, utilidade e condensação são consideradas as principais características de um produto criativo para diversos autores, normalmente usando termos variados para a mesma característica. Assim, o conceito de criatividade de produto se decompõe tipicamente em:

Originalidade: refere-se a um produto com características de surpresa e qualidades embrionárias que influenciam a criação de novos produtos (O'QUIN, 1987). Refere-se, portanto ao grau de novidade de um produto, seja em relação aos seus números, processos, técnicas, materiais e conceitos incluídos (BESEMER; TREFFINGER, 1981).

Utilidade: Mede até que ponto o produto atende às necessidades práticas da situação problemática (O'QUIN, 1987; BESEMER; TREFFINGER, 1981).

Condensação: Considera o nível em que o produto combina os elementos, se o produto foi bem trabalhado e desenvolvido de forma mais completa possível, considerando suas qualidades orgânicas, elegância percebida e complexidade (O'QUIN, 1987; BESEMER; TREFFINGER, 1981).

Assim, de acordo com o foco do presente trabalho, é detalhado o fator de originalidade a partir da seção 2.1.2.

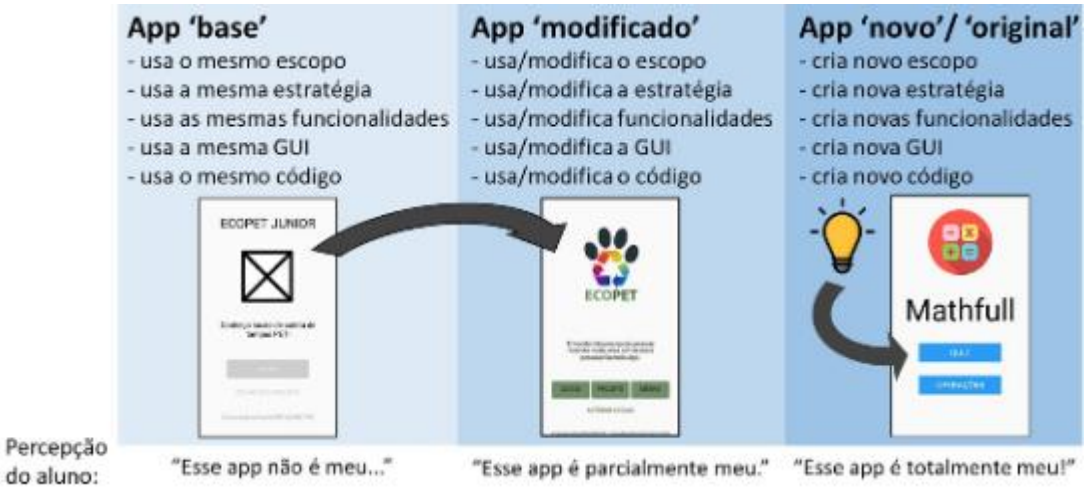
2.1.2 Originalidade no contexto da criatividade no ensino da computação

O ensino de computação na Educação Básica tipicamente foca em ensinar o aluno a desenvolver artefatos computacionais, como, por exemplo, jogos, apps ou até soluções inteligentes (MUSTAFARAJ; TURBAK; SVANBERG, 2017). Este desenvolvimento de artefatos é suportado por ambientes de programação em blocos, como o *Scratch* ou *App Inventor* (MUSTAFARAJ; TURBAK; SVANBERG, 2017) e estão se tornando cada vez mais populares para apresentar e introduzir computação aos iniciantes.

Uma das estratégias para estimular o desenvolvimento da criatividade no ensino de computação na Educação Básica é o ciclo "*Use-Modify-Create*" (LEE *et al*., 2011) guiando a progressão de aprendizagem. No estágio *Use* os alunos tipicamente seguem um tutorial criando um artefato computacional predefinido e não se espera algo criativo deles nesse primeiro estágio. No estágio *Modify* os alunos modificam esse artefato parcialmente, ainda utilizando de funcionalidades do artefato predefinido, mas é esperado que novas características e funcionalidades sejam criadas. Progressivamente, os alunos ganham confiança e avançam para o estágio *Create*, no qual eles são incentivados a desenvolverem seus próprios artefatos computacionais com as características que lhes convém, abordando temas de sua escolha. No estágio *Create* é esperado que o artefato seja completamente novo, sem relação com o artefato base dos estágios anteriores. Assim, existe a oportunidade de os alunos desenvolverem a capacidade de gerar ideias e soluções originais e úteis durante as etapas de modificação e principalmente na criação (WALIA, 2019).

Um exemplo desta progressão é apresentado na Figura 2, onde há os três passos do ciclo "*Use-Modify-Create*" (UMC) detalhadamente.

Figura 2 - Abordagem UMC no contexto de desenvolvimento de apps



Fonte: ALVES et. al, 2020b

Como parte do processo de ensino-aprendizagem, é importante também a avaliação da criatividade pelo desempenho a partir dos artefatos desenvolvidos pelos alunos. Assim, visa-se avaliar como parte da criatividade a originalidade dos artefatos criados pelos alunos.

Mesmo que tipicamente a avaliação da criatividade/originalidade seja feita naturalmente pelas pessoas no dia a dia, no contexto educacional é importante ir além de uma avaliação subjetiva definindo medidas mais objetivas para avaliar a originalidade dos artefatos na sala de aula (MISHRA; HENRIKSEN, 2013). Como esse processo de avaliação requer um esforço considerável por parte do avaliador e também se trata de um julgamento humano, pode ser influenciado por aspectos pessoais e valores culturais (GRESSE VON WANGENHEIM *et al*., 2018b).

Um dos problemas encontrados na avaliação da originalidade do produto é distinguir projetos originais de projetos não originais (MUSTAFARAJ; TURBAK; SVANBERG, 2017). No ensino de computação na Educação Básica, os alunos muitas vezes seguem tutoriais de passo a passo e assim todos acabam desenvolvendo ao final o mesmo artefato, sem originalidade, no nível de *Use*. Por outro lado, nos níveis de *Modify* e *Create* espera-se que os alunos desenvolvam artefatos que variam destes tutoriais predefinidos usando as suas próprias ideias e criatividade para desenvolver o artefato, criando artefatos com originalidade (MUSTAFARAJ; TURBAK; SVANBERG, 2017).

Desta maneira, para avaliar o grau de originalidade de um artefato é importante o **universo de referência**, que podem ser os tutoriais ensinados para os alunos nessas unidades instrucionais ou os artefatos criados por usuários com características semelhantes. Por exemplo, para classificar um artefato como original utilizando um aplicativo base, basta comparar os artefatos criados por usuários com o aplicativo base e inferir se têm as mesmas características (MUSTAFARAJ; TURBAK; SVANBERG, 2017). Entretanto, há outras formas de classificar, como por exemplo, criando um conjunto de vários tutoriais ou artefatos desenvolvidos em um contexto semelhante e comparar o artefato com todos eles (SVANBERG, 2017).

Visando o uso da avaliação para melhoria da aprendizagem do aluno, é fundamental haver uma forma de fornecer um *feedback construtivo* (MISHRA; HENRIKSEN, 2013), incluindo não somente uma pontuação/nota, mas também a identificação dos pontos fortes e fracos além de possíveis sugestões construtivas de como o aluno pode melhorar. Entretanto, o processo de avaliação da criatividade/originalidade é um desafio, pois a criatividade é difícil de medir de forma objetiva com confiabilidade e validade (HENRIKSEN; MISHRA.; MEHTA, 2015; ALVES *et. al*, 2021).

2.2 DESIGN VISUAL DE APLICATIVOS MÓVEIS

O design de interface em aplicativos móveis difere de aplicativos de outras plataformas por ter algumas limitações fisicamente impostas, como tamanho de tela, consumo de energia, etc. Essas características devem ser consideradas para que o design de interface seja eficiente (WASSERMAN, 2010).

O design de interface impacta diretamente na experiência do usuário durante o uso do aplicativo. A forma como é montado o esqueleto e apresentado a distribuição de botões, imagens, textos e outros componentes na interface são fatores importantes para uma experiência positiva de interatividade e usabilidade (LEE *et al.*, 2015; NIELSEN, 1994).

Existem outras dimensões a serem avaliadas para avaliar um aplicativo móvel, conforme mostra a Figura 3. As dimensões e itens são definidos com base no modelo de Garrett (2011) (ALVES; GRESSE VON WANGENHEIM; HAUCK, 2020).

Figura 3 - Rubrica Modifique e Crie com o Universo de Comparação do Produto

Fonte: ALVES; GRESSE VON WANGENHEIM; HAUCK, 2020

A estrutura mostra o grau de diferença entre as entradas e saídas do aplicativo para buscar aplicativos com design de interface semelhante.

O esqueleto define as diferenças entre a organização dos componentes na interface, como navegar entre eles e como as informações do aplicativo são apresentadas na tela e qual grau de diferença tem entre o aplicativo base e o novo aplicativo modificado pelo usuário.

O design de interface envolve o uso de elementos como layout, cor, tipografia e imagens, e é guiado por meta-princípios, incluindo consistência, hierarquia e personalidade (GARRET, 2011; SCHLATTER; LEVINSON, 2013).

Os componentes e os blocos são a programação lógica dos componentes do aplicativo, por meio de funções, estruturas de seleções, laços, entre outros, usados na tela.

Dentre as dimensões apresentadas, o esqueleto e os componentes são o foco do presente trabalho, avaliando a principalmente o design de interface, com ênfase na similaridade entre os componentes, tamanho e posicionamento.

Referente a dimensão de esqueleto, a definição do layout significa posicionar os elementos que compõem a interface seguindo uma certa estrutura, de forma útil e atraente, para que facilite aos usuários o entendimento da interface (SCHLATTER; LEVINSON, 2013). O layout é composto por alguns elementos básicos como: alinhamento, proximidade grid e espaço em branco. Fatores como alinhamento e proximidade afetam diretamente em como o usuário recebe as informações na tela, pois determinam quais componentes estão relacionados entre si.

Os princípios básicos do layout podem ser aplicados independentemente do tamanho da tela, porém, no contexto de aplicativos móveis onde o tamanho da tela é limitado, é fundamental evitar o excesso de informação em uma mesma tela, mostrando apenas as informações essenciais ao usuário.

2.3 ORIGINALIDADE DO DESIGN DE INTERFACE DE APPS

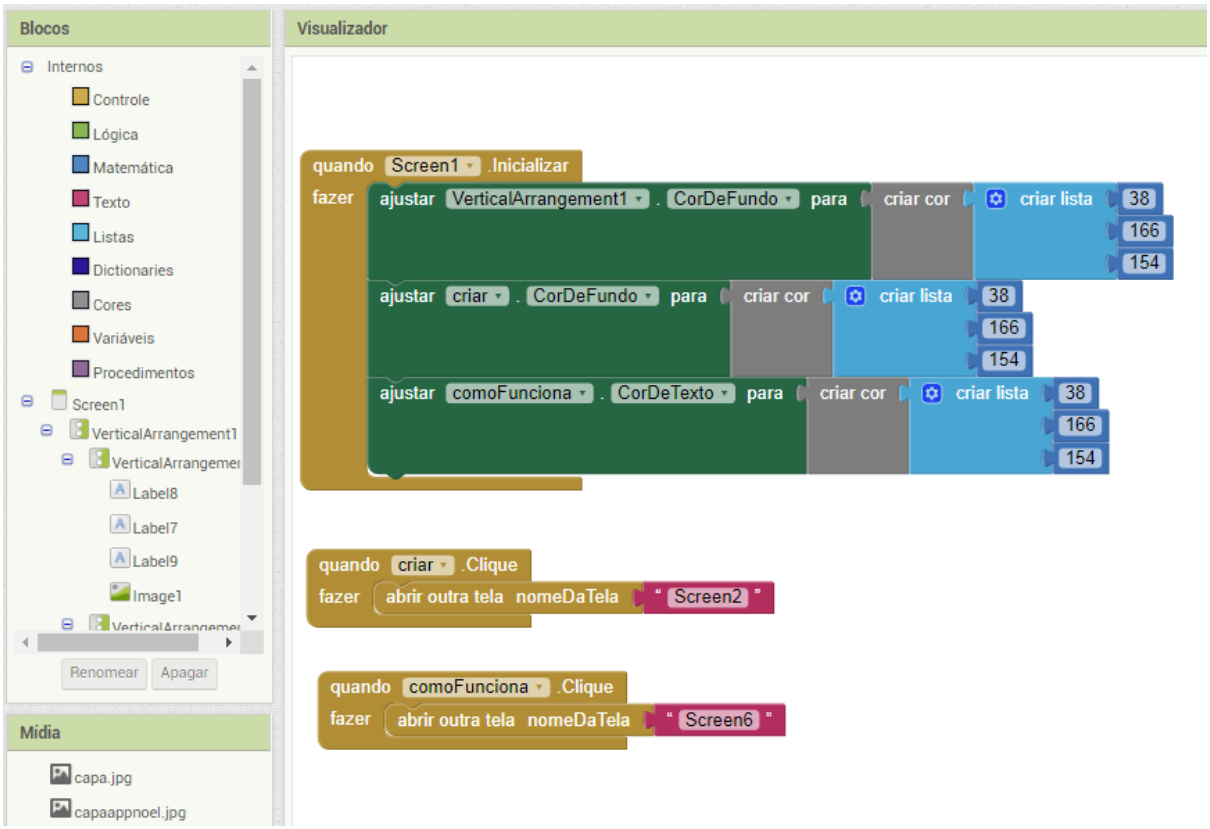
O App Inventor (MIT, 2021) é um ambiente de programação visual amplamente utilizado baseado em blocos para criar aplicativos móveis para dispositivos Android. Ele pode ser facilmente acessado e utilizado gratuitamente por meio de um navegador web. Atualmente, o App Inventor tem mais de 800 mil usuários ativos por mês em 195 países pelo mundo (MIT, 2021).

O App Inventor torna a programação mais acessível a pessoas com pouco ou nenhum conhecimento em programação, pois a aplicação é desenvolvida por meio de blocos visuais de comandos, sem necessidade de compreender e memorizar sintaxes de linguagens textuais (CSTA, 2016).

O desenvolvimento de um aplicativo no App Inventor é dividido em duas partes: Designer e Blocos.

A parte de Blocos suporta/consiste na programação lógica dos componentes do aplicativo, por meio de funções, estruturas de seleções, laços, etc. Os componentes da interface são mostrados no canto esquerdo, onde é possível fazer a interação lógica por meio dos blocos apresentados pelo App Inventor (Figura 4).

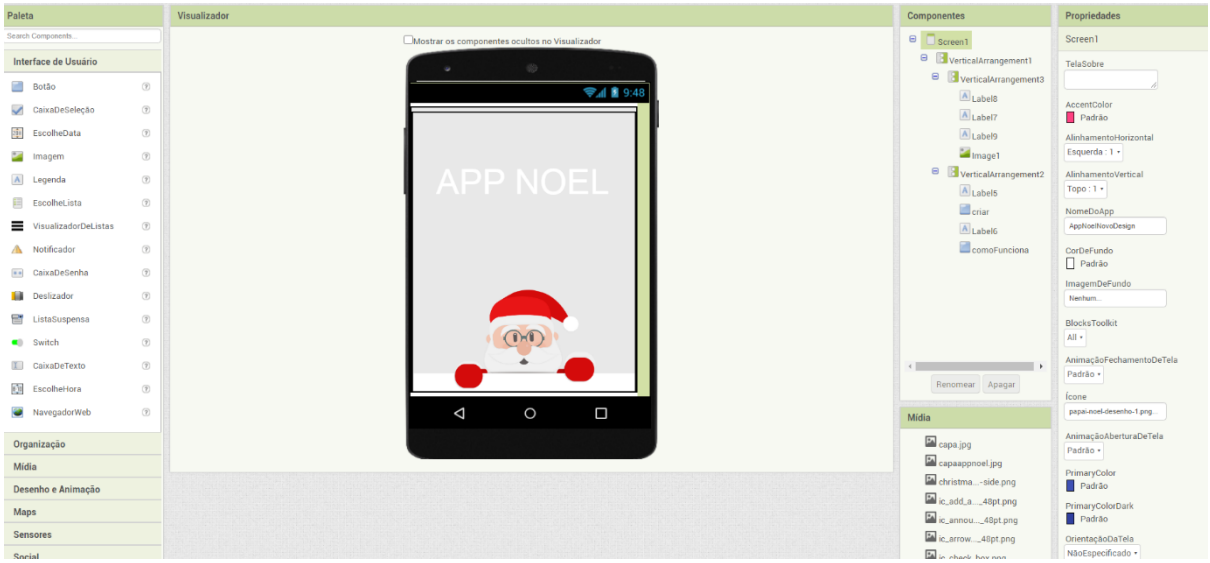
Figura 44 - Tela “Blocos” do App Inventor



Fonte: captura de tela do App Inventor (MIT, 2021).

A parte de Designer permite configurar e arrastar os componentes de interface (Figura 5), como botões, imagens, sensores, sons entre outros. Além disso, todos os componentes podem ter sua aparência configurada da maneira que o usuário desejar, alterando propriedades como tamanho e cor.

Figura 5 - Tela “Designer” do App Inventor



Fonte: captura de tela do App Inventor (MIT, 2021).

O App Inventor disponibiliza diversos componentes para que o usuário utilize durante o desenvolvimento do aplicativo, cada componente possui funcionalidades e características específicas apresentadas no Quadro 1.

Quadro 1 - Componentes de design de interface de usuário disponíveis no App Inventor

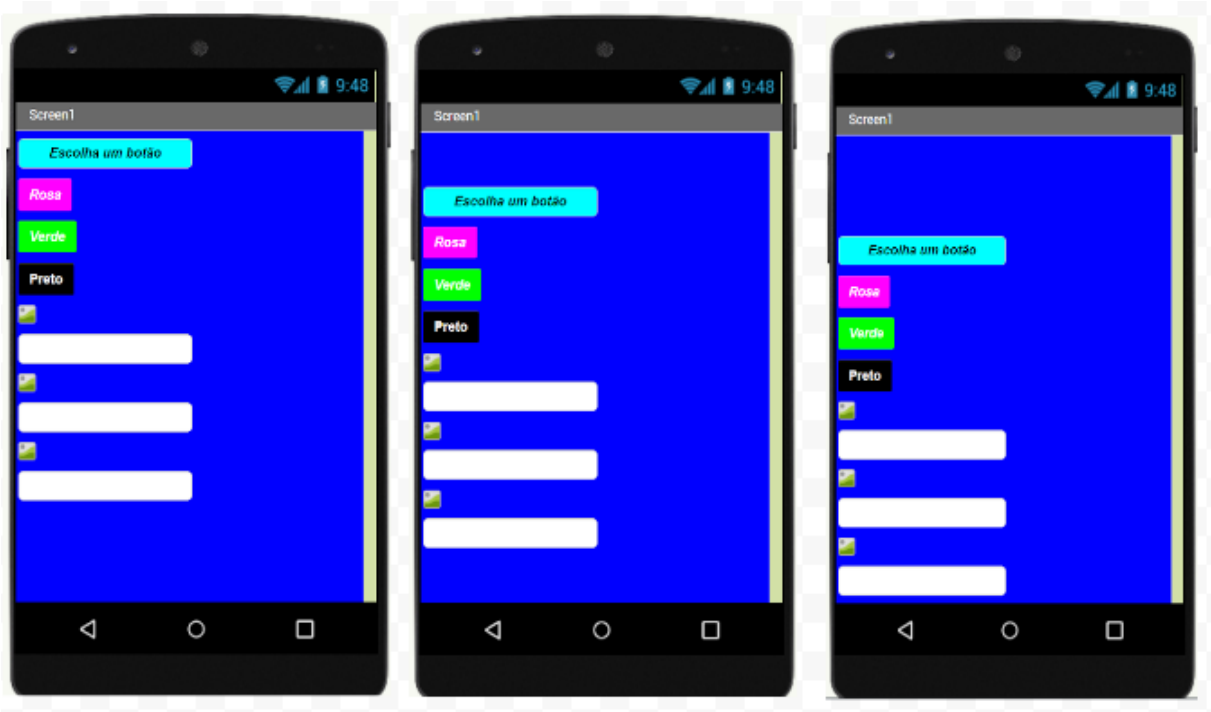
Componente	**Descrição**
Botão	Componente com capacidade de detectar cliques.
EscolheLista	Um botão que, ao ser clicado, exibe uma lista de textos para o usuário escolher.
CaixaDeSeleção	Componente capaz de disparar um evento quando o usuário clica nela.
EscolheData	Um botão que, ao ser clicado, exibe um diálogo para o usuário escolher uma data.
Imagem	Componente para exibição de imagens.
Legenda	Componente para exibição de textos.
VisualizadorDeListas	Componente visível que permite colocar uma lista de elementos de texto para apresentação na tela.
Notificador	Componente que exibe diálogos de alerta, mensagens e alertas temporários, e cria entradas de log Android.
CaixaDeSenha	Uma caixa de texto que esconde o conteúdo dela.
CaixadeTexto	Componente que permite que o usuário insira texto.

Deslizador	Um Deslizador é uma barra de progresso que adiciona um indicador arrastável. O indicador pode ser arrastado para a esquerda ou para a direita para ajustar sua posição.
ListaSuspensa	Um componente para seleção que exibe um pop-up com uma lista de elementos.
Switch	Interruptor que gera um evento ao ser clicado pelo usuário.
EscolheHora	Um botão que, quando clicado, inicia um diálogo que permite ao usuário selecionar um horário.
NavegadorWeb	Componente para visualizar páginas da web.
OrganizaçãoHorizontal	Dispõe outros componentes linearmente, da esquerda para a direita.
HorizontalScrollArrangement	O mesmo que OrganizaçãoHorizontal, com a função adicional de rolamento.
OrganizaçãoEmTabela	Dispõe outros componentes de forma tabular
OrganizaçãoVertical	Dispõe outros componentes linearmente, de cima para baixo, alinhados à esquerda.
VerticalScrollArrangement	O mesmo que OrganizaçãoVertical, com a função adicional de rolamento.
EscolheImagem	Quando clicado, mostra a galeria de imagens do dispositivo e o usuário pode escolher uma imagem.
ReprodutorDeVideos	Reproduz vídeos em uma área retangular na interface.
Pintura	Um painel retangular no qual o desenho pode ser feito e os sprites podem ser movidos.
Mapa	Apresenta um mapa na interface.
EscolheContato	Um botão que, ao ser clicado, exibe uma lista de contatos para o usuário escolher.
EscolheEmail	Uma caixa de texto que sugere contatos quando o usuário começa a digitar um nome ou e-mail.
EscolheNumeroDeTelefone	Um botão que, quando clicado, mostra uma lista dos números de telefone dos contatos para que o usuário escolha um deles.

Fonte: adaptado a partir de MIT, 2021.

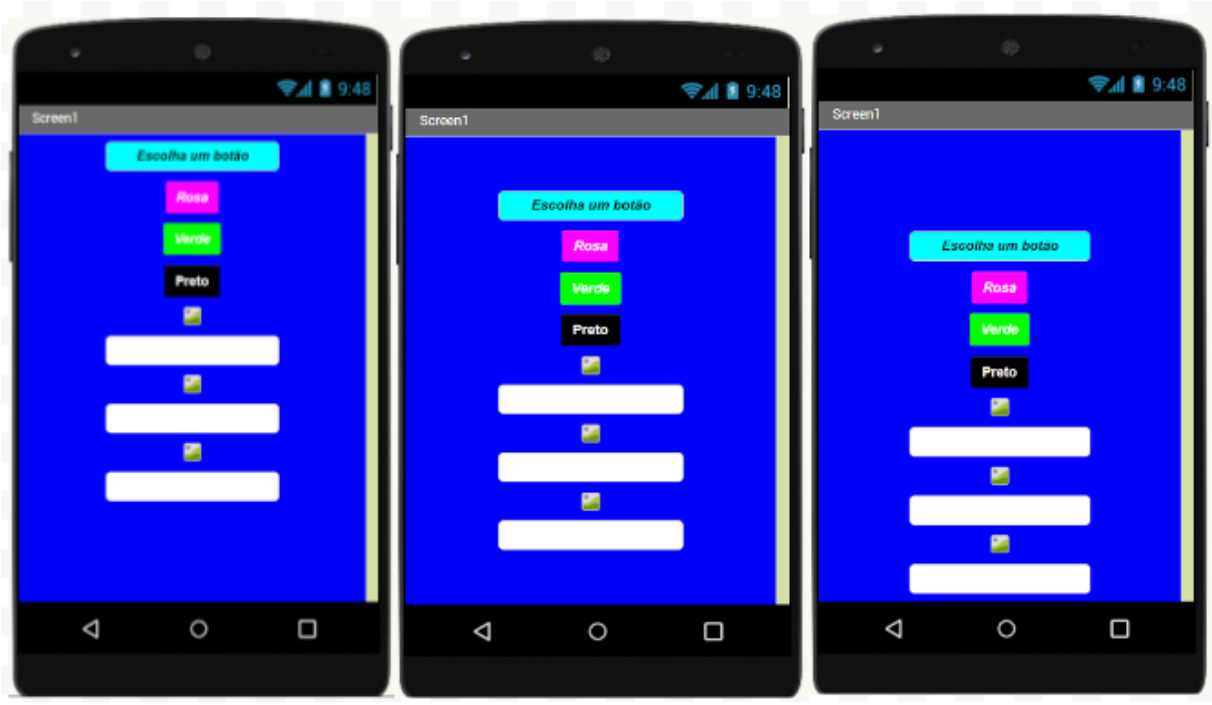
O App Inventor também possibilita a organização do Layout do aplicativo, permitindo alinhar os componentes da interface seguindo uma estrutura, com objetivo de ajudar os usuários a entenderem a interface. Ele permite alinhar os componentes de diferentes maneiras, como apresentado na Figura 6, 7 e 8.

Figura 6 - Diferentes alinhamentos à esquerda para a mesma tela



Fonte: captura de tela do App Inventor

Figura 7 - Diferentes alinhamentos no centro para a mesma tela



Fonte: captura de tela do App Inventor

Figura 8 - Diferentes alinhamentos à direita para a mesma tela

Fonte: captura de tela do App Inventor

Os aplicativos construídos no App Inventor são salvos automaticamente na nuvem, porém é possível ser exportado como um arquivo .aia. Um arquivo .aia é um conjunto de arquivos compactados que incluem arquivos de propriedade do projeto, arquivos de mídia usados pelo aplicativo e, para cada tela do aplicativo, um arquivo .bky e um arquivo .scm. O arquivo .scm contém uma representação JSON de seus componentes enquanto o arquivo .bky uma representação XML com os blocos de programação usados na lógica do aplicativo.

2.4 DEEP LEARNING

John McCarthy, reconhecido como pai da Inteligência Artificial (IA), a IA é “a ciência e a engenharia de criar máquinas inteligentes que tem habilidade para alcançar objetivos da mesma forma que os humanos” (INDIA, 2018). Em resumo, a Inteligência Artificial é a tentativa de representação por máquinas da inteligência humana. Um grande subcampo da área de IA é a *Machine Learning* (ML) que lida com

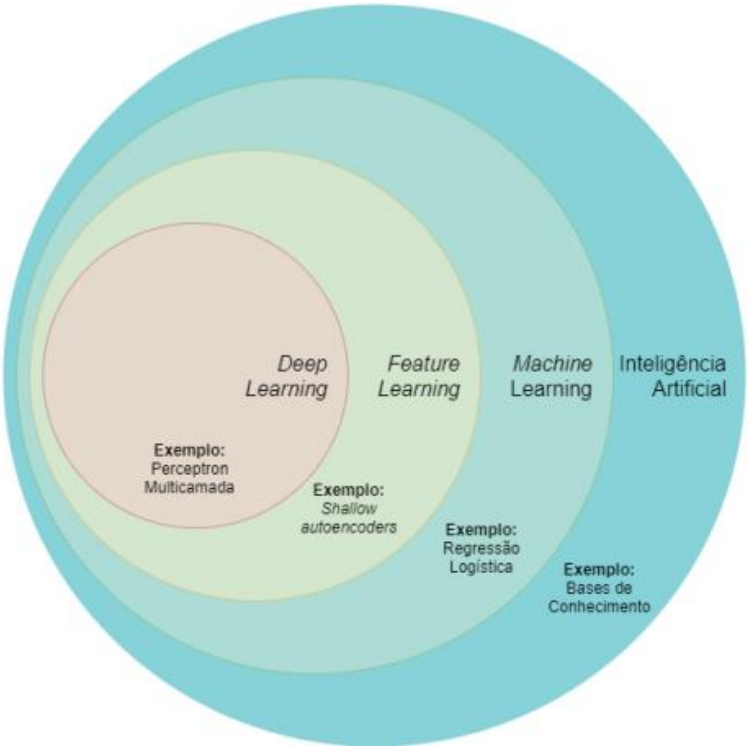
o campo de estudos que dá a computadores a habilidade de aprender sem ser explicitamente programados (SAMUEL, 1959).

Humanos aprendem a identificar objetos recebendo vários exemplos desse objeto e subconscientemente identificando suas características, *Machine Learning* aprende da mesma forma, recebendo constantemente dados e possivelmente tomando decisões futuras a partir deles (SAMUEL, 1959).

Uma área de *machine learning* é a ***Deep Learning***, um tipo de aprendizagem de máquina que permite que modelos computacionais compostos de várias camadas de processamento aprendam representações de dados com vários níveis de abstração (LECUN; BENGIO; HINTON 2015). Estes métodos melhoraram diversas áreas como o reconhecimento de voz, reconhecimento de objeto visual, detecção de objetos e outros domínios, como a pesquisa em remédios e genética (LECUN; BENGIO; HINTON 2015). *Deep Learning* descobre estruturas em grandes conjuntos de dados, trazendo avanço no processamento de imagens, áudio, vídeo e fala (LECUN; BENGIO; HINTON 2015). Isso é possível pois aprende a representar o mundo em termos de uma hierarquia de conceitos, com cada conceito definido por meio de sua relação com conceitos anteriores mais simples, e representações abstratas computadas em termos de outras menos abstratas (GOODFELLOW; BENGIO; COURVILLE, 2017).

A Figura 9 ilustra diferentes disciplinas na área de Inteligência Artificial e mostra que *deep learning* é um tipo de *feature learning*, que por sua vez é um tipo de *machine learning* que é utilizado para muitas, mas não todas, abordagens de IA.

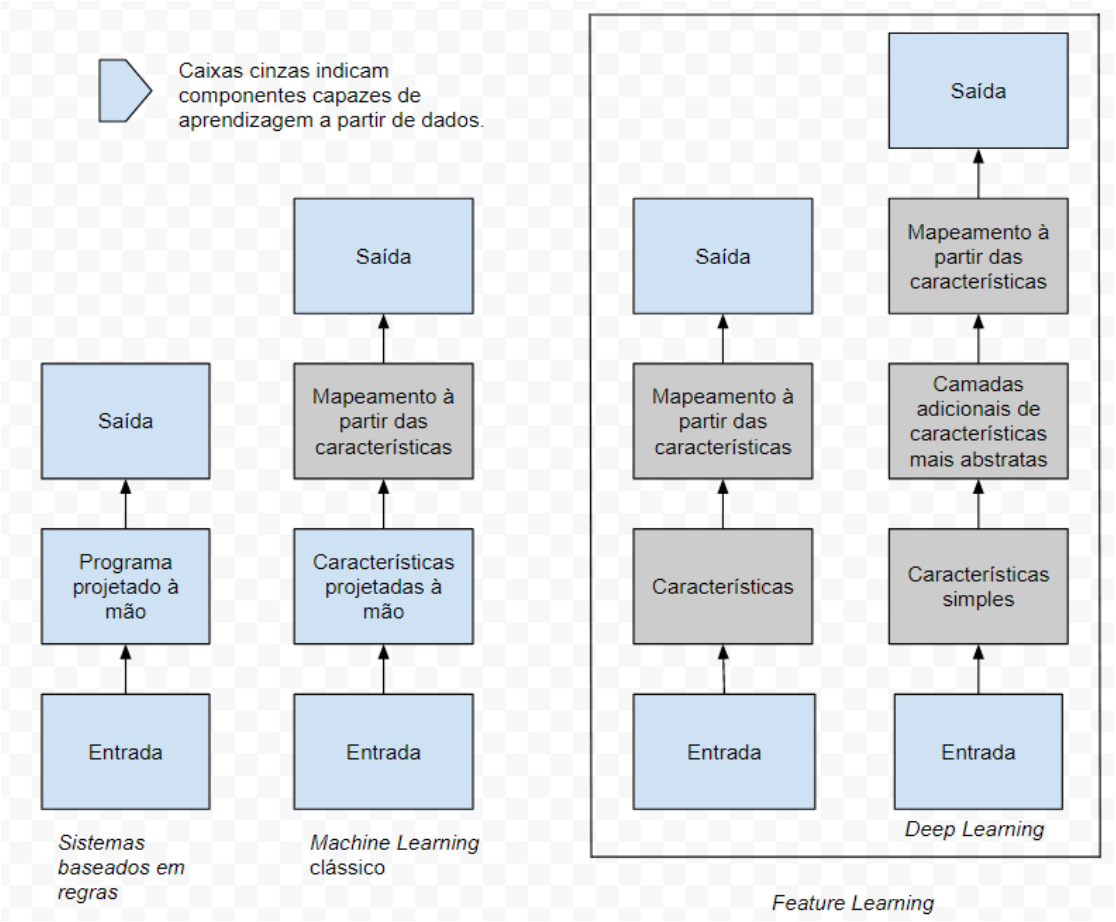
Figura 9 - Relação entre *Deep Learning* e a área de inteligência artificial



Fonte: adaptado de GOODFELLOW; BENGIO; COURVILLE, 2017.

A Figura 10 exibe em alto nível um fluxograma sobre como cada uma funciona, mostrando as similaridades entre elas e como as capacidades de aprendizado a partir de entradas mais simples parecem se tornar maiores na direção de *deep learning*.

Figura 50 - Relação de sistemas de IA em diferentes disciplinas



Fonte: adaptado de GOODFELLOW; BENGIO; COURVILLE, 2017

Os métodos de *deep learning* são métodos de aprendizagem de representação com vários níveis, que permitem serem alimentados com dados brutos e descubra automaticamente as representações necessárias para classificação (LECUN; BENGIO; HINTON 2015). Para isso, *deep learning* utiliza redes neurais de múltiplas camadas.

2.4.1 Redes Neurais

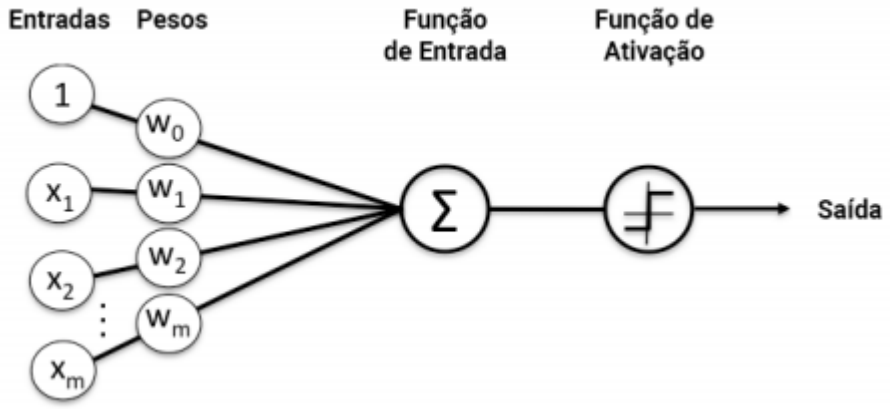
O cérebro humano é um computador altamente complexo, não-linear e paralelo, que tem a capacidade de organizar seus constituintes estruturais, conhecidos por neurônios, de forma a realizar certos processos mais rapidamente que os computadores (RUSSEL; NORVIG, 2009). As redes neurais são um conjunto de

algoritmos, que são projetados para modelar a maneira que o cérebro humano realiza alguma tarefa particular ou função de interesse (PATHMIND, 2020).

As redes neurais interpretam os dados sensoriais e agrupam e categorizam dados brutos. Os padrões reconhecidos por ela são numéricos, armazenados em vetores e para o qual todos os tipos de dados do mundo real devem ser traduzidos (PATHMIND, 2020). Existem diversos tipos de redes neurais (*perceptron, radial basis network, deep feed forward, recurrent neural network, deep convolutional network…)*, cada uma projetada com determinado objetivo, mas nem todas classificam dados.

Uma rede neural é formada por vários nós que são conectados por algum tipo de conexão e cada conexão tem um peso numérico associado a ela (RUSSEL; NORVIG, 2009). Esses pesos são o principal meio de armazenamento de longo prazo em redes neurais e o processo de aprendizado são atualizações destes valores.

Figura 11 - Diagrama exemplo de um nó



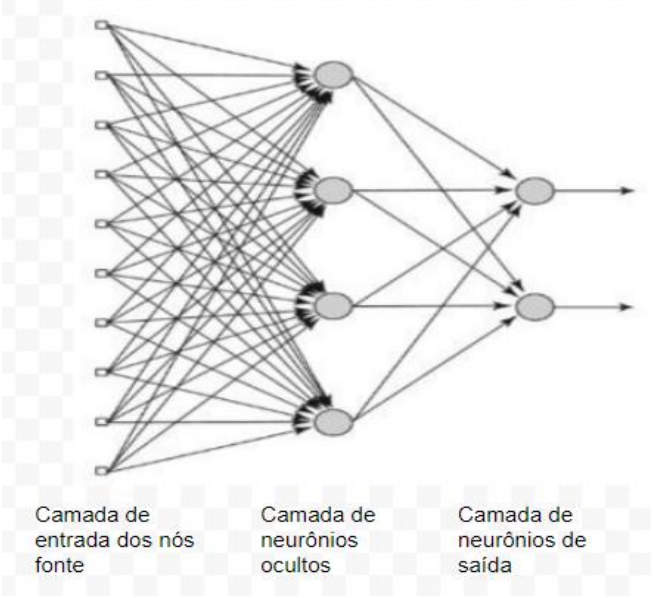
Fonte: adaptado de PATHMIND, 2020.

Cada nodo (Figura 11) então faz uma computação local com a entrada recebida pela função de ativação, que se baseia no somatório dos pesos para ativar ou não um neurônio 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; NORVIG, 2009).

Uma camada é um conjunto dessas unidades que “ligam e desligam” conforme a entrada é alimentada pela rede. A saída de cada camada é a entrada da

camada subsequente, começando por uma camada de entrada inicial que recebe seus dados (PATHMIND, 2020). A exemplificação de uma estrutura de rede neural é apresentada pela figura 12.

Figura 12 - Estrutura básica de uma rede neural

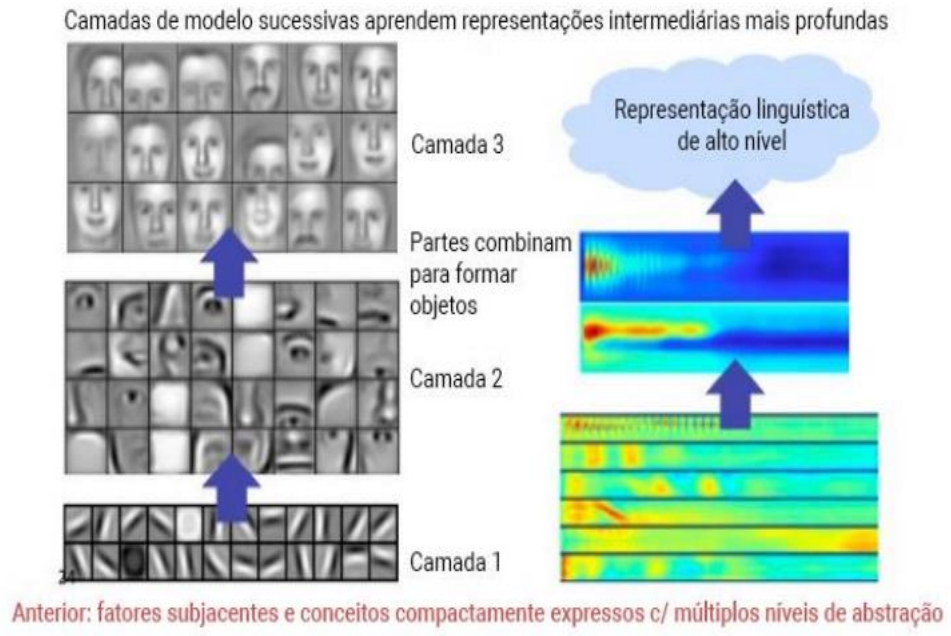


Fonte: HAYKIN, 2007.

2.4.2 *Deep Neural Network (DNN)*

Redes Neurais Profundas (*Deep Neural Networks*) é a estrutura de rede utilizada em *Deep Learning.* Estas redes se distinguem por sua profundidade, ou seja, o número de camadas pelo qual o dado deve passar em um processo de reconhecimento de múltiplas etapas (PATHMIND, 2020). Em redes de *deep learning*, cada camada contém características das camadas anteriores. Quanto mais se avança na rede, mais complexas as características reconhecidas pelos nós, devido a agregação e combinação de características da camada anterior (PATHMIND, 2020).

Figura 13 - Hierarquia de características, com aumento da complexidade e abstração a cada nível



Fonte: adaptado de PATHMIND, 2020.

Essa hierarquia de características, como mostrado na figura 13, torna possível as redes de *deep learning* trabalharem com grandes conjuntos de dados de alta dimensão com bilhões de parâmetros (PATHMIND, 2020). Com isso, essas redes são capazes de descobrir estruturas de dados não-estruturados e não-categorizados, tornando possível trabalhar com a grande maioria dos dados do mundo como áudio, foto, texto e vídeo (PATHMIND, 2020).

Para treinar uma rede neural são utilizados métodos de aprendizado que classificam dados de diferentes maneiras. Os principais métodos de aprendizagem são:

Aprendizagem supervisionada: Um humano classifica um conjunto de dados para que a rede neural tente relacionar os rótulos e os dados do conjunto (PATHMIND, 2020).

Aprendizado não supervisionado: Os dados não são previamente classificados, cabe ao modelo classificar os dados determinando algum padrão entre eles (PATHMIND, 2020).

Durante a criação de uma rede neural, os pesos são inicializados com valores aleatórios e são atualizados ao longo do processo de treinamento por alguns algoritmos, de modo a buscar valores ideais. Para determinar quais valores são melhores, é necessário determinar quão próximo o valor está do valor esperado, isso é geralmente avaliado por uma função de perda (BUSHAEV, 2017).

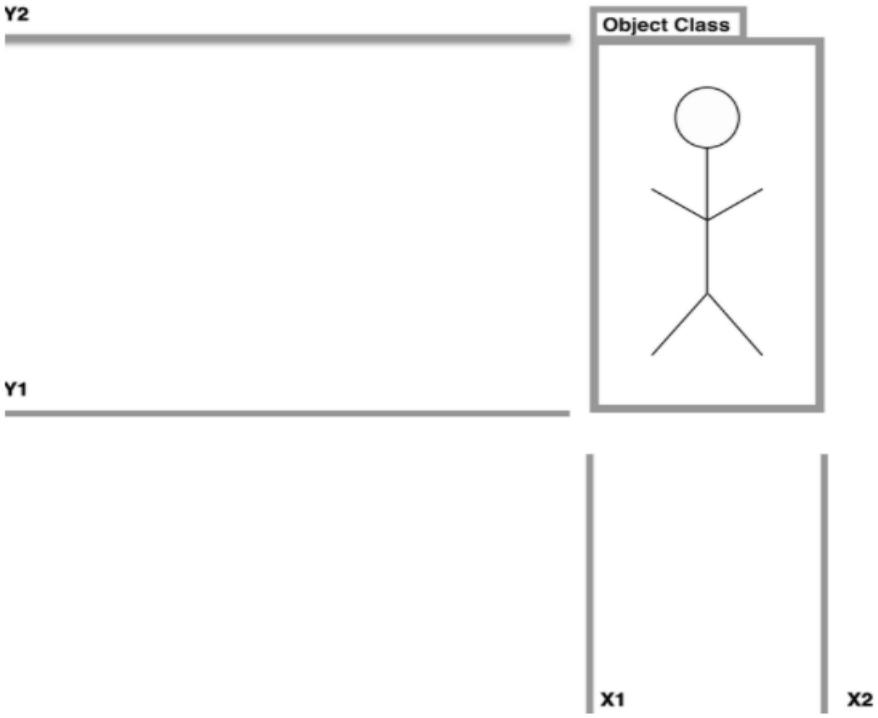
O problema do treinamento é otimizar a função de perda, o que pode ser feito por meio de diversos algoritmos, baseados em gradiente ou não (BUSHAEV, 2017). Portanto, ao longo do treinamento da rede, a rede calcula o valor de perda e ajusta o peso de acordo com o algoritmo escolhido, até que o erro seja minimizado ou o conjunto de dados termine.

2.4.3 *Yolo*

O modelo YOLO (You Only Look Once) é um algoritmo de detecção de objetos em imagem. Ele divide imagens em um sistema de grade. Cada célula na grade é responsável por detectar objetos dentro de si. O YOLO é um dos algoritmos de detecção de objetos mais famosos devido a sua velocidade e precisão (ULTRALYTICS, 2021).

A detecção de objetos ocorre enquadrando o objeto a ser detectado em uma imagem e rotula esses objetos em uma determinada classe conforme mostra a Figura 14.

Figura 14 - Detecção de objetos



Fonte: SOLAWETZ, 2021.

3 ESTADO DA ARTE

3.1 DEFINIÇÃO DO PROTOCOLO DE MAPEAMENTO

O objetivo do presente mapeamento sistemático da literatura é identificar e analisar modelos de análise automatizada da originalidade de design de interfaces de usuário de apps, focando em soluções que adotam técnicas de IA. O mapeamento segue o processo proposto por Petersen *et al*. (2008) e Petersen, Vakkalanka e Kuzniarz (2015) e visa responder à seguinte pergunta de pesquisa: Quais modelos/ferramentas existem para automaticamente avaliar a originalidade do produto a partir de design de interface de usuário de apps Android (criados com o App Inventor no contexto da Educação Básica) usando técnicas de IA?

A pergunta de pesquisa é refinada nas seguintes perguntas de análise:

PA1. Quais modelos/ferramentas de avaliação da originalidade do design de interfaces de apps Android existem?

PA2. Quais as características do modelo de avaliação da originalidade do design de interfaces?

PA3. Como foi feita a preparação do conjunto de dados e a rotulação?

PA4. Como o modelo/ferramenta foi desenvolvido?

PA5. Como a qualidade do modelo/ferramenta foi avaliada?

Critérios de inclusão e exclusão:

Os artigos relevantes foram selecionados conforme os seguintes critérios de inclusão e exclusão:

Incluir artigos científicos e artefatos que apresentam algum modelo de avaliação da originalidade em relação ao design de interface de usuário de aplicativos móveis;

São considerados somente modelos que adotam o uso de inteligência artificial;

Incluir apenas artigos em inglês no período de 2007, ano do lançamento do primeiro smartphone (MURTAZIN, 2010), até 2021 acessíveis via Portal CAPES;

Incluir artigos que apresentam modelos de avaliação do design visual, não limitando a abordagens para uso educacional para se ter uma visão mais ampla do estado da arte.

Critérios de qualidade: Foram incluídos apenas artigos contendo informações substanciais indicando como o design visual é avaliado de acordo com sua originalidade. Ou seja, os critérios de avaliação devem estar explícitos.

Bases de dados: As buscas foram realizadas nas principais bases de dados e bibliotecas digitais da área da computação: ACM Digital Library, IEEE Xplore Digital Library, ScienceDirect, Scopus, arXiv, e Google Scholar. Pesquisamos no Google Scholar pois é considerado aceitável como uma fonte adicional visando a minimização do risco de omissão por buscar artigos de forma mais ampla de diversas áreas de conhecimento (PIASECKI; WALIGORA; DRANSEIKA, 2018).

String de Busca: Com base na pergunta de pesquisa, buscas informais foram utilizadas para calibrar a string de busca, a qual foi definida com base nos termos relevantes da pergunta de pesquisa do mapeamento, resultando nos termos de busca, sinônimos e traduções apresentados na Quadro 2.

Quadro 2 - Termos de busca relevantes, sinônimos e tradução (inglês)

Termo	**Sinônimos**	**Tradução (inglês)**
Aplicativos móveis	Aplicativo, Android, App Inventor	*Mobile application, Android, App Inventor, App*
Design Visual	Design de interface de usuário	Visual Languages, UI design, User Interface
Criatividade	Originalidade, Similaridade, novidade	*Creativity, Originality, Similarity, Novelty*
Deep Learning	Inteligência artificial	*Deep learning, Artificial Intelligence*

Fonte: o autor.

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

("*Mobile Application" OR "Android" OR "App Inventor"* OR "App") AND (*"User Interface*" OR "*UI Design*" OR "*Visual Languages*") AND ("*Creativity"* OR *"Original"* OR *"Similarity" OR "Novel"*) AND ("*Deep Learning"* OR "*Artificial Intelligence"*)

Após a definição da string de busca, foi realizada a sua adaptação para cada base de dados utilizando a linguagem específica de cada uma (Quadro 3).

Quadro 3 - Strings de busca utilizadas nas diferentes bases de dados

Base de dados	***String* de busca**
ACM	[[All: "mobile application"] OR [All: "android"] OR [All: "app inventor"] OR [All: "app"]] AND [[All: "user interface"] OR [All: "ui design"] OR [All: "visual languages"]] AND [[All: "creativity"] OR [All: "original"] OR [All: "similarity"] OR [All: "novel"]] AND [[All: "deep learning"] OR [All: "artificial intelligence"]] AND [Publication Date: (01/01/2007 TO 04/30/2021)]
arXiv.org	Query: order: -announced_date_first; size: 50; include_cross_list: True; terms: AND all=("Mobile application" OR "Android" OR "App Inventor" OR "App") ; AND all= ("User interface" OR "UI design" OR "Visual Languages") ; AND all= ("Creativity" OR "Original" OR "Similarity" OR "Novel"); AND all= ("Deep learning" OR "Artificial Intelligence")
Google Scholar	(“Mobile Application” OR “Android” OR “App Inventor”) AND (“User Interface” OR “UI Design” OR “Visual Languages”) AND (“Creativity” OR “Original” OR “Similarity”) AND (“Deep Learning” OR "Artificial Intelligence")
IEEE	("Mobile application" OR "Android" OR "App Inventor" OR "App") AND ("User interface" OR "UI design" OR "Visual Languages") AND ("Creativity" OR "Original" OR "Similarity" OR "Novel") AND ("Deep learning" OR "Artificial Intelligence")
Sciencedirect	("Android” OR “App Inventor") AND ("User interface” OR “Visual Languages”) AND (“Original” OR "Similarity") AND ("Deep learning” OR “Artificial Intelligence" OR "Machine Learning") Year: 2007-2021
SCOPUS	("Mobile Application" OR "Android" OR "App Inventor" OR "App") AND ("User Interface" OR "UI Design" OR "Visual Languages") AND ("Creativity" OR "Original" OR "Similarity" OR "Novel") AND ("Deep Learning" OR "Artificial Intelligence") AND PUBYEAR > 2006 AND (LIMIT-TO (LANGUAGE,"English"))

Fonte: o autor.

3.2 EXECUÇÃO DE BUSCA

A busca dos artigos foi realizada em abril de 2020 pelo autor do presente trabalho em conjunto com outra pesquisadora do GQS e revisada pela orientadora. A busca inicial resultou em 19.738 artigos, dos quais foram selecionados artigos relevantes de acordo com os critérios de inclusão, exclusão e qualidade (Quadro 4).

Quadro 4 - Resultados da busca

Base de Dados	**Quantidade de artigos resultantes da busca**	**Quantidade de artigos analisados**	**Quantidade de artigos potencialmente relevantes (título e abstract)**	**Quantidade de artigos relevantes (com base no artigo na íntegra)**
IEEE Xplore Digital Library	10	10	1	1
ACM Digital Library	944	150	9	4
Scopus	3.814	150	22	12
arXiv.org e-print archiv	4	4	2	1
GoogleScholar	14.200	150	6	3
ScienceDirect	766	150	3	0
Total				11

Fonte: o autor.

3.3 ANÁLISE DE RESULTADOS

Para responder à questão de pesquisa, as informações relevantes às perguntas de análise foram extraídas dos 11 artigos relevantes encontrados e são apresentadas a seguir:

3.3.1 Quais modelos/ferramentas de avaliação da originalidade do design de interfaces de apps Android existem?

No total, foram encontradas 11 publicações que contém alguma forma de avaliação da originalidade do design de interfaces de apps Android ou App Inventor. Essas publicações são listadas no Quadro 5.

Quadro 5 - Documentos resultantes da execução de busca.

Citação	Referências Bibliográficas
(GE, 2019)	GE, X. Android GUI search using hand-drawn sketches. *In*: **Proc. of the 41st International Conference on Software Engineering: Companion**, IEEE Press, p. 141–143, 2019.
(BEHRANG; REISS; ORSO, 2018)	BEHRANG, F., REISS, S.P., ORSO, A. GUIfetch: supporting app design and development through GUI search. **Proc. of 5th International Conference on Mobile Software Engineering and Systems**, ACM, New York, USA, p. 236-246, 2018.
(MUSTAFARAJ; TURBAK; SVANBERG, 2017)	MUSTAFARAJ, E.; TURBAK, F.; SVANBERG, M. Identifying Original Projects in App Inventor. **Proc. of Florida Artificial Intelligence Research Society Conference**, Massachusetts, USA, maio. 2017.
(DEKA *et al.,* 2017)	DEKA, B. et al. Rico: A Mobile App Dataset for Building Data-Driven Design Applications. **Association for Computing Machinery**, New York, NY, USA, p. 845–854, 2017.
(XIE; LIN; XU, 2019)	XIE, Y.; LIN, T.; XU., H. User Interface Code Retrieval: A Novel Visual-Representation-Aware Approach. *In:* **Proc. of IEEE Access**, 7, 162756-162767, 2019.
(SVANBERG, 2017)	SVANBERG, M. Using feature vector representations to identify similar projects in app inventor. In: **Proc. Of IEEE Blocks and Beyond Workshop**, p. 117–18, 2017
(TURBAK *et al.*, 2017)	TURBAK, F. *et al.* Work in progress: Identifying and analyzing original projects in an open-ended blocks programming environment. In: **Proc. of the 23rd Int. Conference on Distributed Multimedia Systems, Visual Languages and Sentient Systems**, p. 115-117, 2017.
(CHEN *et al.,* 2020)	CHEN, J. et al. Wireframe-based UI Design Search through Image Autoencoder. **ACM Trans. Softw. Eng. Methodol.**, v. 29, n. 3, jul. 2020.
(CHEN, X. *et al,* 2019)	CHEN, X. *et al.* Recommending software features for mobile applications based on user interface comparison. **Requirements Engineering**, v. 24, n. 4, p. 545-559, 2019.

(NGUYEN; VU; PHAM, 2018)	NGUYEN, T., VU, H., PHAM. Deep Learning UI Design Patterns of Mobile Apps. *In:* **Proc. of IEEE/ACM 40th International Conference on Software Engineering: New Ideas and Emerging Technologies Results** (ICSE-NIER), p. 65-68, 2018.
(HU *et al*., 2020)	HU, Y. et al. Robust App Clone Detection Based on Similarity of UI Structure, *In*: **Proc. of IEEE Access**, 8, 77142-77155, 2020.

Fonte: o autor.

3.3.2 Quais as características do modelo de avaliação de originalidade do design de interfaces?

O foco dos critérios de avaliação do design de interface varia entre as abordagens encontradas. Portanto, as abordagens foram classificadas em uma das seguintes categorias (Quadro 6):

Estrutura: Avalia a similaridade utilizando como base os *sketches* (esboços) para buscar aplicativos com design de interface semelhante comparando com os esboços.

Componentes/Blocos: Avalia a similaridade utilizando como base os componentes e blocos, que são a programação lógica dos componentes do aplicativo, por meio de funções, estruturas de seleções, laços, entre outros, usados na tela.

Quadro 6 - Tipo dos critérios de design interface em cada abordagem

Autoria	**Tipo dos Critérios**	
	Estrutura	**Blocos/Componentes**
(GE, 2019)	x	
(NGUYEN; VU; PHAM, 2018)	x	
(BEHRANG; REISS; ORSO, 2018)	x	
(MUSTAFARAJ; TURBAK; SVANBERG, 2017)		x
(DEKA *et al*., 2017)	x	
(XIE; LIN; XU, 2019)	x	
(SVANBERG, 2017)		x
(MUSTAFARAJ et al., 2017b)		x

(HU *et al*., 2020)	x	
(CHEN *et al,* 2019)		x
(CHEN *et al.,* 2020)	x	

Fonte: o autor.

A maior parte das abordagens (7 de 11) avaliam critérios relacionados à estrutura da interface de usuário. Critérios desse tipo avaliam o posicionamento dos componentes disponibilizados na tela e a forma como estão organizados, utilizando *sketches* como base e buscando em um conjunto de dados interfaces de usuário semelhantes.

Outras abordagens (4 de 11) avaliam critérios relacionados aos blocos e componentes utilizados na interface de usuário. Critérios desse tipo avaliam os blocos e componentes não pelo seu valor, mas pelo seu tipo. Intuitivamente, a originalidade de um projeto é capturada pelo tipo e número de blocos que não são comuns em projetos não originais.

3.3.3 Como foi feita a preparação do conjunto de dados e a rotulação?

Analisando os artigos relevantes, foi possível perceber a ausência de um conjunto de dados conhecido, com muitos dos artigos criando seu próprio conjunto de dados ou utilizando repositórios públicos (Quadro 7).

Quadro 7 - Dados extraídos para responder à Pergunta de Análise 4

Citação	**Dados**				
	Nome do Conjunto de dados	**Rotulação**	**Quantidade e instâncias**	**Data de criação**	**Referência**
(GE, 2019)	Rico	Comparando Árvores rotuladas	72.000	NI	NI
(BEHRANG; REISS; ORSO, 2018)	Utiliza repositórios públicos	Automatizado utilizando GUIfetch	+300.000	NI	www.github.com e https://bitbucket.org/
(MUSTAFARAJ; TURBAK; SVANBERG, 2017)	Conjunto de dados próprio	Elementos anotados manualmente	902	2015	http://appinventor.mit.edu
(DEKA *et al*., 2017)	Rico	NI	72k	NI	NI
(XIE; LIN; XU, 2019)	Conjunto de dados próprio	Comparando Árvores rotuladas	6.750	NI	NI

(SVANBERG, 2017)	Conjunto de dados próprio	Elementos anotados manualmente	894	2015	NI
(TURBAK *et al.*, 2017)	Utiliza repositórios públicos	Elementos anotados manualmente	NI	NI	NI
(CHEN *et al.,* 2020)	NI	Não utiliza rotulação	NI	NI	NI
(CHEN *et al,* 2019)	Utiliza repositórios públicos	NI	61.089 páginas de IU coletadas de 10.477 aplicativos	NI	NI
(NGUYEN; VU; PHAM, 2018)	Utiliza repositórios públicos	Elementos anotados manualmente	NI	NI	NI
(HU *et al*., 2020)	Conjunto de dados próprio	Elementos anotados manualmente	404.650	NI	NI

Fonte: o autor

Mustafaraj, Turbak e Svanberg, (2017) utilizaram um conjunto de dados dos projetos de alunos do curso que os autores ministraram em outono de 2015, Svanberg (2017) utilizou parte desse mesmo conjunto de dados acrescentando mais alguns projetos.

Turbak *et al*. (2017) utilizaram dois conjuntos de dados de projetos de usuários do App Inventor: O primeiro se originou entre dezembro de 2013 e fevereiro de 2017, que inclui projetos de mais de 10.000 usuários aleatoriamente. O segundo contou com projetos de 46.320 usuários e estima-se que cada usuário criou 20 ou mais projetos.

Como existem conjunto de dados com screenshots de Apps disponíveis para o uso Ge (2019) e Deka *et al.* (2017) se basearam em um desses, o conjunto de dados Rico.

Outras abordagens como Behrang, Reiss e Orso (2018), Nguyen, Vu e Pham (2018), Chen *et al.* (2019) utilizaram repositórios públicos disponibilizados gratuitamente.

Das publicações encontradas, (5 de 11) fazem a rotulagem manualmente, (2 de 11) utilizam árvores rotuladas para comparação entre o sketch e a UI encontrada.

Além disso, (1 de 11) não utiliza a rotulagem e (1 de 11) utiliza uma ferramenta que faz isso automaticamente, porém sem especificar como.

3.3.4 Como o modelo/ferramenta foi avaliado?

Algumas abordagens tais como Ge (2019), Behrang, Reiss e Orso (2018), Xie, Lin e Xu (2019), partem de um *sketch* feito à mão e utilizam técnicas de *deep learning* para buscar em um conjunto de dados as UIs similares. A partir disso, é calculada uma pontuação de similaridade entre o *sketch* e todas as UIs encontradas.

Ge (2019), faz isso por meio de um *framework* utilizando *deep learning,* que faz a comparação das árvores entre o esboço e os esqueletos GUI correspondentes, utilizando o algoritmo *largest weighted common subtree embeddings* (LWCSE).

Xie, Lin e Xu (2019) converteram o esboço em uma árvore de representação visual para então treinar o modelo CNN com muitos esboços rotulados.

Behrang, Reiss e Orso (2018) apresentaram um modelo que foi treinado para automaticamente comparar a pontuação de similaridade entre o sketch e a UI correspondente.

Nguyen, Vu e Pham (2018) utilizaram linguagem natural como entrada para retornar UIs mais simples e menos elaborados para depois gerar UIs com design mais profissional. Para isso, foram utilizadas redes neurais recorrentes e redes adversárias geradoras, que são treinadas para aprender os padrões de design de milhões de UIs de apps móveis.

Algumas abordagens como Mustafaraj, Turbak e Svanberg (2017), Svanberg (2017) e Turbak *et al.* (2017), utilizam conjuntos de dados próprios e rotularam manualmente as UIs em original ou não original. Após isso, criaram vetores de recursos dos projetos e utilizaram várias métricas como Euclidiana, Cosine e Jaccard para calcular a similaridade. Dessas abordagens, Mustafaraj, Turbak e Svanberg (2017) utilizou *clustering* e métodos de agrupamento hierárquico com base em semelhanças entre vetores de recursos para classificar automaticamente os projetos como não originais ou originais.

Para calcular a similaridade da IU por meio das melhores correspondências de componentes de duas IUs, Chen *et al* (2019) utilizam a similaridade de texto para medir a similaridade dos componentes da IU. É utilizado o modelo *latent dirichlet allocation* (LDA) para modelar um documento em uma mistura de palavras. Após isso é utilizado 2 ferramentas, uma em inglês e outra em chinês para calcular a similaridade

de texto e então é utilizado um algoritmo genético para otimizar a eficiência do tempo de determinar as correspondências quase ótimas de componentes.

Chen *et al*. (2020) utilizaram técnicas de *deep learning* para treinar um *wireframe* utilizando um grande conjunto de dados de design UI sem rótulos.

Observa-se que em geral, as abordagens seguem uma metodologia de treinar um modelo utilizando um grande conjunto de dados para então fazer uma comparação de similaridade. No entanto, algumas das publicações não relatam o processo de desenvolvimento.

3.3.5 Como a qualidade do modelo foi avaliada?

A avaliação de cada abordagem é apresentada após o Quadro 8.

Quadro 8 - Dados extraídos para responder à Pergunta de Análise 5

Citação	**Dados**		
	Medidas de desempenho	**Avaliação do modelo**	**Resultados da avaliação**
(GE, 2019)	Ranking de similaridade do APK utilizado como base	Seis APKs do repositório escolhidos como base e desenhados a mão	APK base costuma aparecer entre os 6 resultados mais similares
(BEHRANG; REISS; ORSO, 2018)	RQ1: Precisão do GUIFetch. RQ2: Quão relevante são as recomendações. RQ3: Quão bem o GUIFetch responde ao ranqueamento do usuário. RQ4: Quanto às recomendações ajudam.	RQ1: Tendo um *sketch* e uma série de apps, um deles corresponde ao sketch o modelo indicou corretamente o app semelhante. RQ2: Entrevista com participantes prototipando 35 telas e indicando a relevância da sugestão. RQ3: Utilizaram correlações de ranqueamento clássicas (Kendall's Tau e Spearman) para avaliar o ranqueamento dos usuários e tiveram uma comprovação das respostas dos usuários. RQ4: Entrevista com 16 participantes desenhando um app de *mobile bank* ou de *address book* gerando 34 telas	RQ1: Apontou corretamente o app mais parecido com o *sketch* de teste e teve correlações com as respostas dos usuários bem próximas das respostas oferecidas por eles. RQ2: 54% das recomendações foram consideradas 100% relevantes e outros 23% com uma relevância maior que 80%. RQ3: 14 telas tiveram correlação total em ambas métricas e outras 10 tiveram resultado entre 0.6 e 0.867 por Kendall's Tau e 0.7 à 0.943 por Spearman. RQ4: Resultado mostra que o método ajuda a gerar ideias, apenas um usuário não utilizou de nenhuma forma as

			recomendações fornecidas
(MUSTAFARAJ; TURBAK; SVANBERG, 2017)	Calcular a precisão de cada rótulo individualmente	876 projetos	Em uma certa distância de Jaccard, a precisão é de 89%
(DEKA *et al*., 2017)	NI	NI	NI
(XIE; LIN; XU, 2019)	RQ1: Avaliação da similaridade e o reuso do código gerado. RQ2: Comparação com outros métodos com outros métodos de similaridade entre árvores.	RQ1: 15 participantes foram escolhidos para avaliar cerca de 6750 telas geradas utilizando *intraclass correlation* coefficient para verificar se as respostas dos avaliadores condizem com o espero e foi visto RQ2: Calculado o erro quadrático médio e o coeficiente de correlação de Pearson entre o método criado, tree edit distance, sequências multidimensionais e Rule	RQ1: O *intraclass correlation coefficient* girou em torno de 0.725 e 0.889 com uma média de 0.822 ($p < 0.0001$). RQ2: Teve um resultado 50% menor do erro quadrático médio frente ao melhor dos outros métodos e o coeficiente de correlação de Pearson de 0.883, com o segundo modelo tendo 0.692.
(SVANBERG, 2017)	Utilização de técnicas para comparar características de blocos e componentes	894 projetos	A maioria das características está nos blocos. No entanto, vetor de componentes podem ser úteis para distinguir design de tela
(TURBAK *et a*l., 2017)	Utilização de técnicas de agrupamento para encontrar projetos originais ou não originais	902 projetos	Classifica um conjunto de dados similares
(CHEN *et al*., 2020)	RQ1: Efetividade de cada modelo. RQ2: Precisão do mecanismo de busca. RQ3: Capacidade de generalização. RQ4: Estudo do usuário	54.987 Screenshots de UI	Comparação entre as técnicas utilizadas
(CHEN *et al*., 2019)	RQ1: A precisão dos cálculos de similaridade da IU RQ2: Recomendação de recurso	RQ1: 30 UIS representativas, para cada UI, 10 UIS do repositório são selecionadas do repositório aleatoriamente RQ2: Métricas de taxa de acerto e MRR	No geral, as métricas retornam bons resultados, indicando que o método obtém um bom desempenho
(NGUYEN; VU; PHAM, 2018)	NI	NI	NI
(HU *et al*., 2020)	Quão eficiente é a abordagem	1.777 apps	Abordagem consegue detectar diferentes tipos de clones de apps

Fonte: o autor

Algumas publicações não relataram as avaliações dos modelos.

Ge (2019) realiza uma avaliação simples, escolhendo seis screenshots da sua base de dados, fazendo sketches com base nos screenshots escolhidos e verificando

em que posição o screenshot original ficou na lista de screenshots visualmente mais similares com o sketch retornado pelo sistema (Figura 15), concluindo que geralmente o sistema obtém performance adequada, com todos os *screenshots* originais entre os 6 primeiros resultados.

Figura 15 – Exemplo avaliação

UI scenario	Target App package name	Similarity score*	Rank*
login & register	com.choiceoflove.dating	81.08%	1
search	com.parfield.prayers.lite	64.86%	5
setting	com.google.android.apps.messaging	72.97%	5
list	com.baviux.pillreminder	67.57%	6
picture	com.kudago.android	78.42%	1
detail	com.google.android.play.games	70.27%	1

* Scores and ranks of the six target GUIs in the final search result list, respectively.

Fonte: GE, 2019.

Behrang, Reiss e Orso (2018) utilizaram 16 participantes com conhecimento prévio em design de aplicativo Android ou experiência em desenvolvimento de aplicativo Android. Os participantes primeiro desenham um esboço, utilizam do GUIfetch para receber as recomendações e então classificam tais recomendações como relevantes ou não e decidem se aplicam.

Mustafaraj, Turbak e Svanberg (2017) realizaram uma classificação por agrupamento utilizando a métrica de distância Jaccard generalizada e então classificando em original ou não original baseado na distância. Turbak *et al*. (2017) também realiza os mesmos passos, porém, quer distinguir também projetos criados em determinados horários para saber se alunos estão realizando o mesmo curso.

Svanberg (2017) utiliza as métricas de distância Jaccard e Cosine para classificar os projetos similares por componentes e blocos.

Xie, Lin e Xu (2019) utilizaram especialistas para avaliar os pares de interface, e para evitar a influência das cores no julgamento dos participantes, descoloriram todas as interfaces de usuário.

Para avaliar o cálculo da similaridade, Chen *et al*. (2019) calcularam valores AP (precisão média) para cada conjunto de dados e MAP (precisão média das médias)

para todo o conjunto. O valor MAX é 0,45 e representa o valor ideal nessa abordagem, Chen *et al*. (2019) chegou à valores de 0,457 de AP e 0,417 de MAP que está muito próximo do valor ideal.

Chen *et al.* (2020) avaliam quão bem cada um dos três modelos representa os componentes e a efetividade do mecanismo de busca, bem como a avaliação de seu mecanismo de busca, por meio da opinião de especialistas.

Hu *et al.* (2020) realizam uma avaliação por vetores de recursos as características similares em pares de apps para detectar clones de apps.

3.4 DISCUSSÃO

Os resultados revelam que ainda são recentes as abordagens de avaliação de aplicativos móveis Android que incluem a avaliação da similaridade do design visual. Diversos trabalhos encontrados têm um foco parcialmente similar ao objetivo do presente trabalho. A maioria avalia a similaridade utilizando como base a estrutura do *layout* do aplicativo e baseado em *sketches*. Mustafaraj, Turbak e Svanberg (2017), Svanberg (2017) e Turbak *et al.* (2017) são os únicos que utilizam de alguma maneira apps do App Inventor e que avaliam os critérios especificamente pelos tipos de blocos e componentes.

Os trabalhos encontrados utilizam diferentes conjuntos de dados, sendo maioria os que utilizam um conjunto de dados próprios e repositórios públicos. Apenas Ge (2019) e Deka *et al.* (2017) utilizam o conjunto de dados Rico. Para assegurar uma rotulação balanceada, a maioria dos trabalhos adota um procedimento de rotulação dos elementos manualmente. Outros (GE, 2019; XIE; LIN; XU, 2019) comparam árvores rotuladas e (DEKA *et al*., 2017) não forneceu informações sobre esta questão.

Os trabalhos encontrados demonstram grande variação nos modelos de *machine learning* utilizados, apontando o grande número de possíveis abordagens ao problema em questão. Além disso, apenas Mustafaraj, Turbak e Svanberg (2017), Svanberg (2017) e Turbak *et al.* (2017) efetivamente trabalham com apps do App Inventor, foco do presente trabalho, e de alguma forma utilizam técnicas de aprendizado não-supervisionado, como *clustering*.

Muitos dos trabalhos selecionados focam no processo de desenvolvimento da solução, e não em detalhes do modelo resultante, resultando na falta de detalhes na parte de avaliação do desempenho.

De forma geral, os trabalhos encontrados não focaram tanto na avaliação e divulgação dos resultados, dificultando a comparação entre os modelos encontrados.

3.4.1Ameaças à validade

Como em qualquer mapeamento sistemático, existem possíveis ameaças à validade dos resultados. Mapeamentos sistemáticos podem ter resultados tendenciosos pois resultados positivos têm maior chance de serem publicados do que resultados negativos. Devido à pouca influência que a tendência destes artigos tem sobre este mapeamento sistemático, uma vez que se refere a uma avaliação do estado da arte, este viés não representa uma ameaça grave.

Outro risco é o de omitir estudos relevantes. Para mitigá-lo, a *string* de busca foi construída buscando utilizar todos os termos relevantes e incluindo sinônimos. Além disso, a busca foi realizada em diversas bases de dados científicas, reduzindo as chances de omitir estudos relevantes.

Para mitigar riscos na seleção dos estudos e extração de dados, foi utilizada uma definição detalhada dos critérios de inclusão e exclusão. Foi definido e documentado um protocolo rígido para a seleção do estudo, que foi realizada e revisada cuidadosamente. Além disso, a seleção dos trabalhos relevantes foi discutida com outros pesquisadores do GQS, INCoD, INE, UFSC e revisada pela orientadora até um consenso ser atingido.

4 MODELO DE AVALIAÇÃO DA ORIGINALIDADE DE DESIGN DE INTERFACE DE APLICATIVOS ANDROID

Este capítulo apresenta a proposta do modelo para a avaliação da originalidade de design de interface de aplicativos Android.

4.1 ANÁLISE DE REQUISITOS

O objetivo deste trabalho é o desenvolvimento de um modelo para avaliar a originalidade do esqueleto de design de interface de aplicativos Android criados com App Inventor no nível *create* do ciclo *"Use-Modify-Create"* (LEE *et al.,* 2011).

O modelo recebe como entrada um conjunto de *screenshots* de telas de um app "novo" criado por um estudante como resultado de processo de aprendizagem com App Inventor e como resultado indica uma nota de originalidade do esqueleto do design de interface desse app em comparação com um universo de referência. O universo de referência é composto de um conjunto de apps criados com App Inventor em contextos semelhantes e para os quais já foram automaticamente detectados os elementos e o layout das telas.

O processo é dividido nos seguintes passos:

1. Inicialmente é selecionado um conjunto de *screenshots* para ser utilizado como dados para treinamento e validação e é feita a rotulação dos componentes desses *screenshots* manualmente.
2. É feito o treinamento da rede com base nesses *screenshots* já rotulados.
3. É realizada a detecção automática dos elementos (posição, tamanho e tipo do elemento de UI) nos *screenshots* do universo de referência do novo app usando Deep Learning.
4. Comparação dos elementos e layout das telas do app novo com o universo de referência identificando um grau de similaridade par a par com todos os apps no universo.
5. Cálculo de uma nota de originalidade no intervalo de [0…10] com base no grau de similaridade.

A Figura 16 exemplifica o fluxo de trabalho idealizado.

Figura 16 - Fluxo de trabalho idealizado

Fonte: o autor.

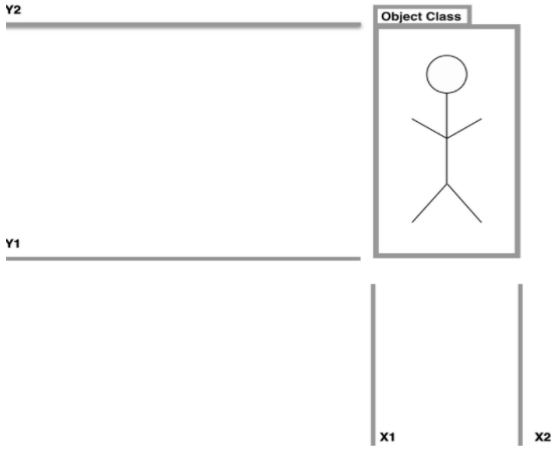
4.2 CRIAÇÃO DE UM MODELO DE DETECÇÃO DE ELEMENTOS E LAYOUT DE UI

4.2.1 Detecção e seleção de componentes

Para detectar automaticamente os componentes e layout de UI em *screenshots* de apps foi adotada a técnica de detecção de objetos utilizando Deep Learning. Foi utilizado YOLO sendo uma das principais arquiteturas atualmente para detecção de objetos (ALVES, 2020). Escolheu-se o modelo YOLOv5 que é a versão mais atualizada do YOLO no momento (ULTRALYTICS, 2021). E, dentro da família dos modelos se usou a versão YOLOv5s por ser a versão atual e por motivos tecnológicos, pois os outros modelos exigem mais memória e CUDA (*Compute Unified Device Architecture*) para treinar o modelo e são muito mais lentos para executar o treinamento.

A detecção de objetos do YOLOv5 enquadra o componente a ser detectado (Figura 17) e gera informações em .txt com seu tipo, tamanho e posição.

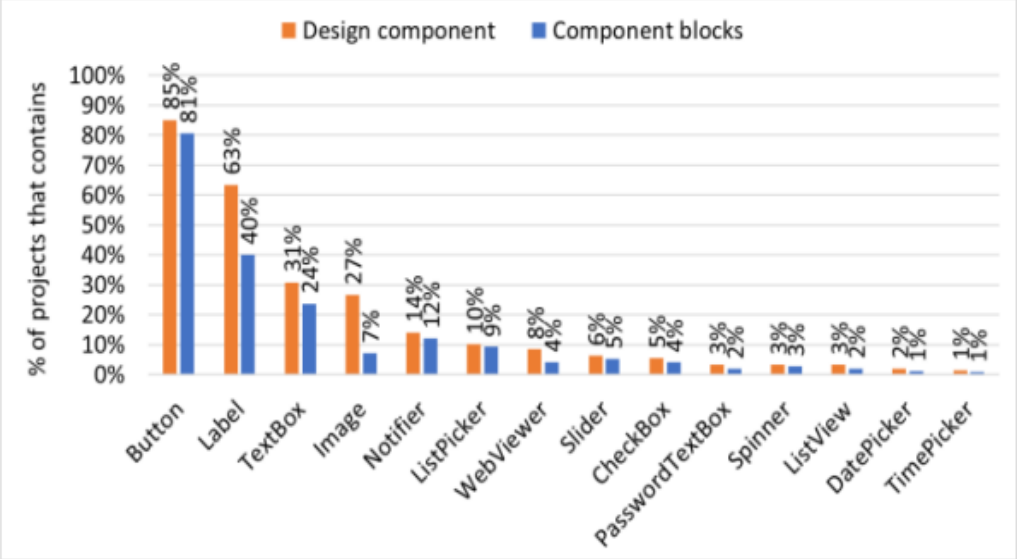
Figura 17 - Detecção de objetos

	Retornando um .txt com 5 colunas e *n* linhas no seguinte formato: 6 0.207407 0.64974 0.385185 0.125521 onde cada coluna representa: *class, center_x, center_y, width, height*

Fonte: SOLAWETZ, 2021.

Devido ao grande número de componentes de interface de usuário disponibilizados pelo App Inventor, foram selecionados aqueles que aparecem com frequência maior em aplicativos do App Inventor, com base em Alves, Gresse Von Wangenheim e Hauck (2020) (Figura 18).

Figura 18 - Frequência de uso dos componentes de interface de usuário no App Inventor



Fonte: ALVES; GRESSE VON WANGENHEIM; HAUCK., 2020.

Porém, considerando o foco do presente trabalho em relação aos componentes visuais, observa-se que alguns dos elementos não aparecem visualmente na tela (Notificador) e outros são visualmente idênticos (Botão, VisualizadorDeListas, EscolheData, EscolheHora). Sendo mais recentes, alguns elementos visualmente importantes como Mapa, Deslizador, não foram considerados no levantamento feito por Alves et al. (2020). Com base nessa análise foram selecionados um conjunto de componentes considerados relevantes para o presente trabalho, conforme mostra ao Quadro 9.

Quadro 9 - Componentes consideradas na avaliação

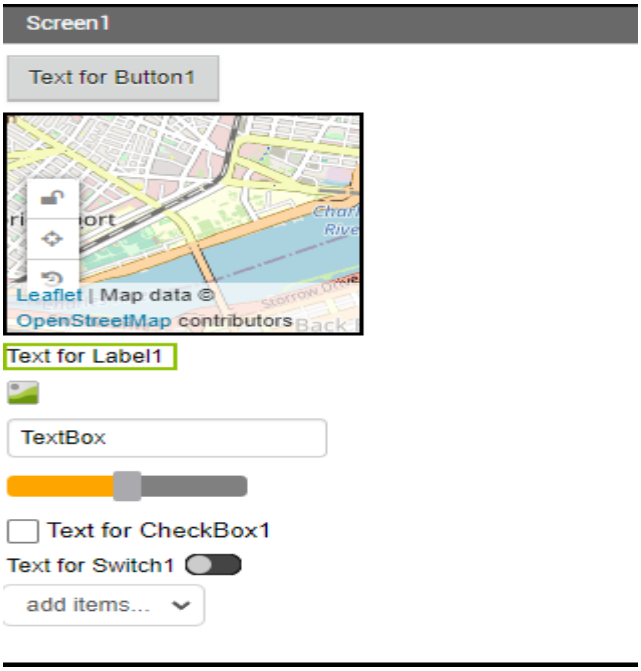
Componente
Botão
Mapa
Legenda
Imagem
CaixaDeTexto
Deslizador
CaixaDeSeleção

Switch
EscolheLista

Fonte: Elaborado pelo autor.

A representação de todos esses componentes selecionados no App Inventor é demonstrada na Figura 19.

Figura 19 – Visualização dos componentes selecionados no App Inventor

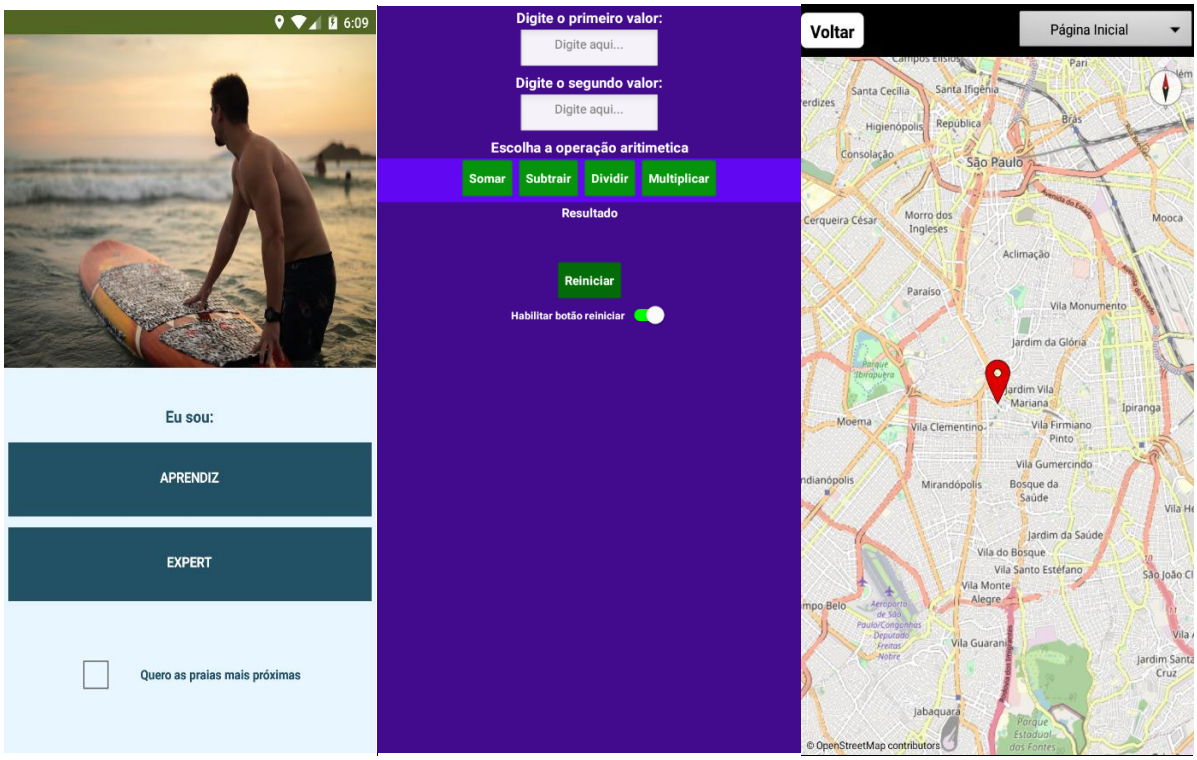


Fonte: Captura de tela do App Inventor (MIT, 2021)

4.2.2 Preparação do conjunto de dados

Coleta de dados: Levando em consideração o foco do presente trabalho em apps criados com App Inventor, foi criado um conjunto de dados incluindo *screenshots* de apps criados com App Inventor. Esses *screenshots* foram coletados de 1.551 apps selecionados aleatoriamente de um conjunto de mais de 88 mil apps da galeria do App Inventor ou criados no contexto da iniciativa Computação na Escola. A Figura 20 representa alguns exemplos destes *screenshots* selecionados.

Figura 20 – Exemplos de *screenshots* selecionados

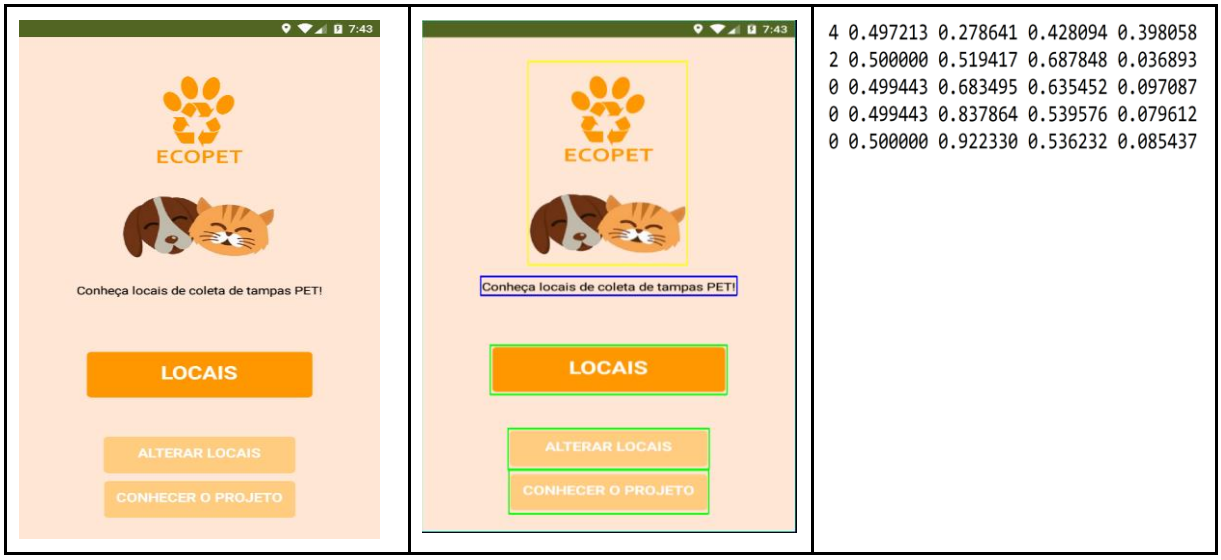


Fonte: captura de tela (2022)

No total, foram coletados 8.432 screenshots de interface de aplicativos de App Inventor, foram utilizados 292 para o treinamento e 73 para validação. Os outros 8.067 foram utilizados para a criação do universo de referência.

Rotulação dos dados: Para o treinamento do modelo foi adotado a aprendizagem supervisionada, criando manualmente as anotações referentes ao screenshot. As anotações foram criadas utilizando a ferramenta YoloLabel (https://github.com/developer0hye/Yolo_Label). Visualmente quando a anotação de uma imagem é feita por meio do YoloLabel, é apresentado o bounding box conforme mostrado na Figura 21, com os “retângulos” em volta dos componentes capturados, então, essa ferramenta de rotulação gera como saída um arquivo .txt no formato para o modelo Yolo, com os seguintes atributos: class, center_x, center_y, width, height identificando as anotações com as classes de componentes de UI e as posições dos componentes (Figura 21).

Figura 21 – Exemplo de *screenshot* e arquivo com o conjunto de *labels*

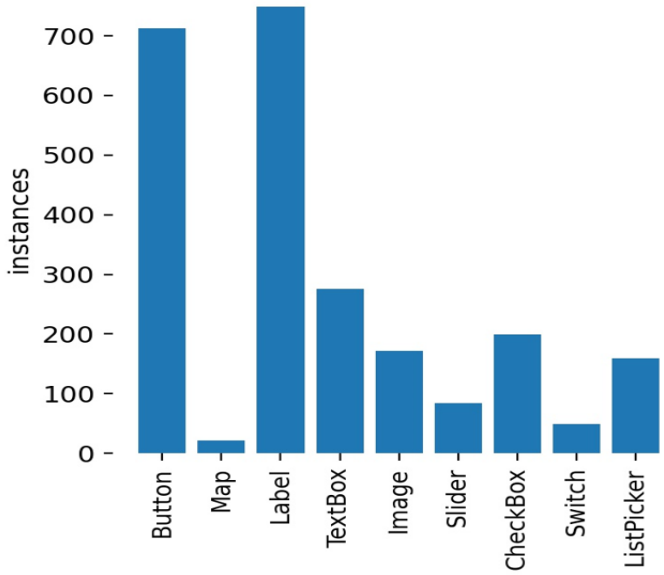


Fonte: captura de tela (2022)

Dos 8.432 *screenshots* coletados, 300 foram selecionados aleatoriamente e manualmente rotulados por pesquisadores da iniciativa Computação na Escola.

Analisando a distribuição da aparência dos componentes UIs nas telas, observou-se que alguns elementos aparecem com uma frequência muito diferente dos outros (Figura 22).

Figura 22 – Frequência dos componentes de interface no conjunto de dados



Fonte: dados da pesquisa (2022)

Nota-se que há uma grande diferença na frequência dos componentes, principalmente os elementos botões e legendas. Foi realizada uma busca em apps com os elementos menos frequentes, mas como esses apps também contém os outros elementos mais frequentes, a inclusão dessas telas não ia alterar significativamente essa distribuição. Assim, como a distribuição corresponde com a frequência de uso dos componentes (Figura 22) se manteve o conjunto de dados criados.

4.2.3 Treinamento do modelo de detecção de componentes

Seleção do Modelo:

Foi utilizado o modelo YOLO (*You Only Look Once*) de detecção de objetos em imagem. O YOLO é um algoritmo de detecção de objetos que divide imagens em um sistema de grade. Cada célula na grade é responsável por detectar objetos dentro de si. O YOLO é um dos algoritmos de detecção de objetos mais famosos devido à sua velocidade e precisão (ULTRALYTICS, 2021).

Para o presente trabalho, foi selecionado o YOLOv5s por ser a versão atual e por motivos tecnológicos, pois os outros modelos exigem mais memória e CUDA para treinar o modelo e são muito mais lentos para executar o treinamento.

Treinamento de modelo:

O treinamento foi realizado utilizando pesos pré-treinados disponibilizados pelo próprio YOLOv5 e adaptando os arquivos necessários, modificando o número de classes, epochs e batch_size conforme o Quadro 10. Os passos para o treinamento do modelo estão disponibilizados no Apêndice A.

Quadro 10 - Parâmetros utilizados no treinamento do modelo

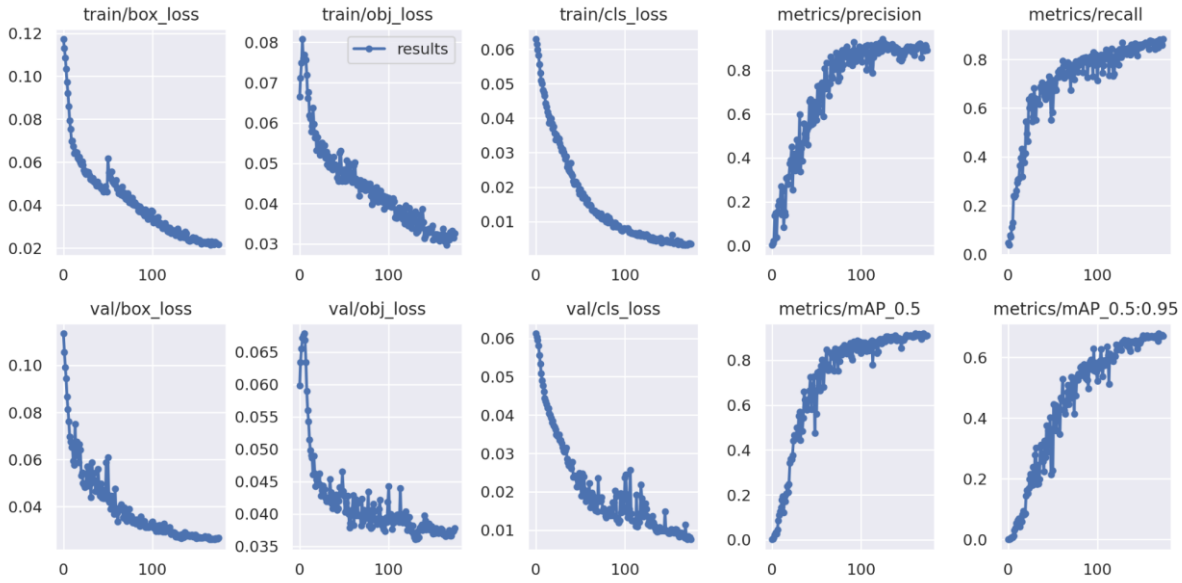
Quantidade de épocas	O treinamento foi iniciado com 300 épocas, porém, como após 175 épocas não foi identificado melhora do desempenho do modelo, a quantidade de épocas foi reduzida para 175 para evitar o overfitting.
Tamanho do *batch*	32
Quantidade total de	365 imagens

imagens usadas para treinamento e validação	
Divisão do conjunto de dados	O conjunto de dados foi dividido aleatoriamente em um conjunto de treinamento com 292 imagens (80%) e um conjunto de validação com 73 imagens (20%)

Fonte: o autor.

O treinamento da rede foi realizado utilizando um Jupyter Notebook no Google Colab. Na avaliação do treinamento do modelo de detecção de objetos várias medidas foram analisadas, disponibilizadas pelo próprio YOLOv5 (Figura 23).

Figura 23 - Métricas de desempenho do treinamento da rede



Fonte: dados da pesquisa (2022).

Pode-se notar um alto crescimento em todas as métricas conforme aumenta o número de iterações, aumenta a precisão da classificação e diminui as taxas de erros.

4.2.4 Avaliação do desempenho do modelo de detecção de objetos

Para avaliar o desempenho na detecção de objetos, a medida mais comum é a Precisão Média (*Average Precision* - AP) e para comparar o desempenho entre

componentes é utilizado o AP médio - mAP (*Mean Average Precision*) (ZOU *et al*. 2019). A Figura 24 mostra os melhores valores do mAP total e por componente após o treinamento da rede com 200 épocas.

Figura 24 – Visão geral das medidas de desempenho

Model Summary: 213 layers, 7034398 parameters, 0 gradients, 15.9 GFLOPs

Class	Images	Labels	P	R	mAP@.5	mAP@.5:.95: 100% 2/2 [00:02<00:00, 1.12s/it]
all	73	644	0.901	0.873	0.914	0.678
Button	73	171	0.9	0.899	0.942	0.767
Map	73	5	0.744	0.8	0.799	0.798
Label	73	224	0.834	0.924	0.902	0.565
TextBox	73	99	0.981	0.818	0.924	0.706
Image	73	41	0.909	0.805	0.843	0.582
Slider	73	21	0.941	0.76	0.92	0.517
CheckBox	73	32	0.966	0.906	0.957	0.724
Switch	73	15	0.952	1	0.995	0.608
ListPicker	73	36	0.884	0.944	0.946	0.836

Fonte: dados da pesquisa (2022).

Os valores do mAP são considerados em intervalos de confiança de IOU (*Intersection over Union*), que calcula um valor para saber se a *bounding box* prevista é compatível com a *bounding box* previamente rotulada (ZOU et al, 2019). Usa-se tradicionalmente 0.5, com uma taxa de sobreposição de 50%, que indica o quanto a *bounding box* prevista é compatível com a *bounding box* previamente rotulada (quanto elas se sobrepõem) e 0.5:0.05:0.95 significa a partir de IOU = 0.5, com passos de 0.05, até um IOU de 0.95. Isso resulta em 10 valores de APs, então é calculada uma média para fornecer um valor único.

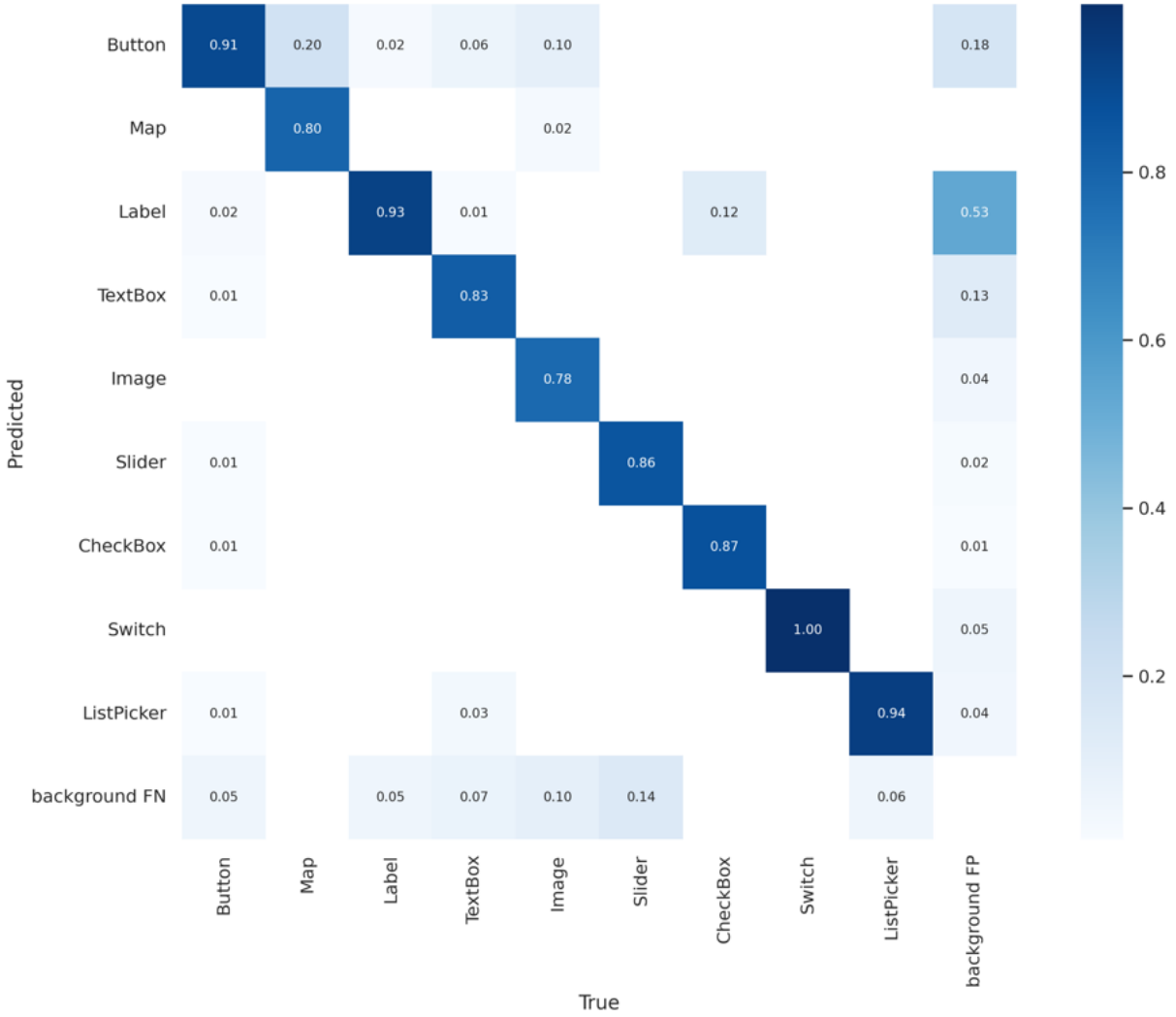
Nota-se que o mAP (IOU = 0.5) atinge uma precisão de 91.4% que é excelente. O mAP (IOU 0.5:0.95) atinge uma precisão de 67.8% o que é aceitável pois é calculado com um *threshold* de valores entre 0.50 até 0.95.

Analisando os componentes individualmente pode se observar que, de forma geral, o mAP@.5 de todos os componentes está acima de 0.8, com a maioria até acima de 0.9. Somente o mAP (IOU = 0.5) do componente mapa é 0.79, provavelmente pelo conjunto de testes conter poucos componentes desses tipos (somente 5 Mapas). Porém, mesmo assim ainda representa um resultado aceitável. Outro componente (Imagem) também é o único com um mAP (IOU = 0.5) abaixo de

0.9 provavelmente porque a forma como alguns apps são modelados com uma lista de botões com cores torna difícil para a rede discernir entre botão/imagem, como será exemplificado na página 62.

Outro ponto positivo é que mesmo em relação aos componentes com características semelhantes (Botão, EscolheLista e CaixaDeTexto) o modelo treinado é capaz de classificar os componentes com alta precisão (Figura 25).

Figura 25 – Matriz de confusão do modelo treinado



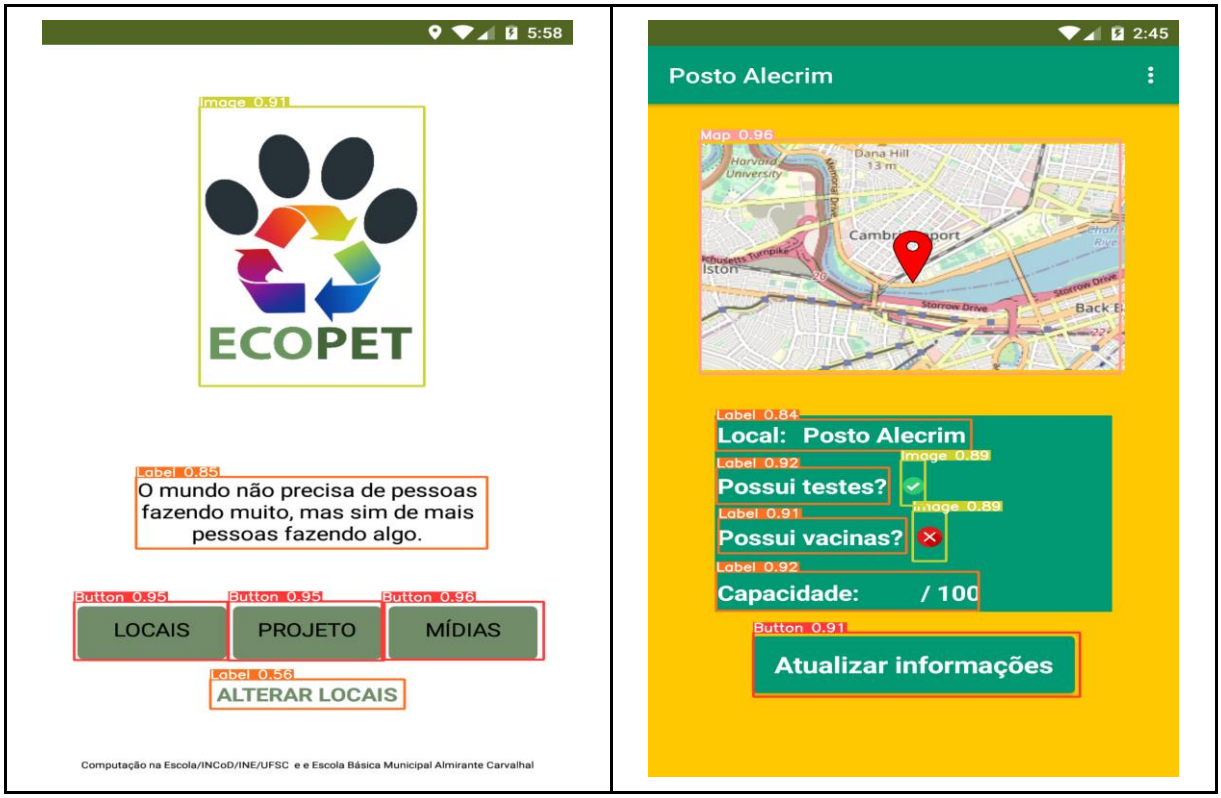
Fonte: dados da pesquisa (2022).

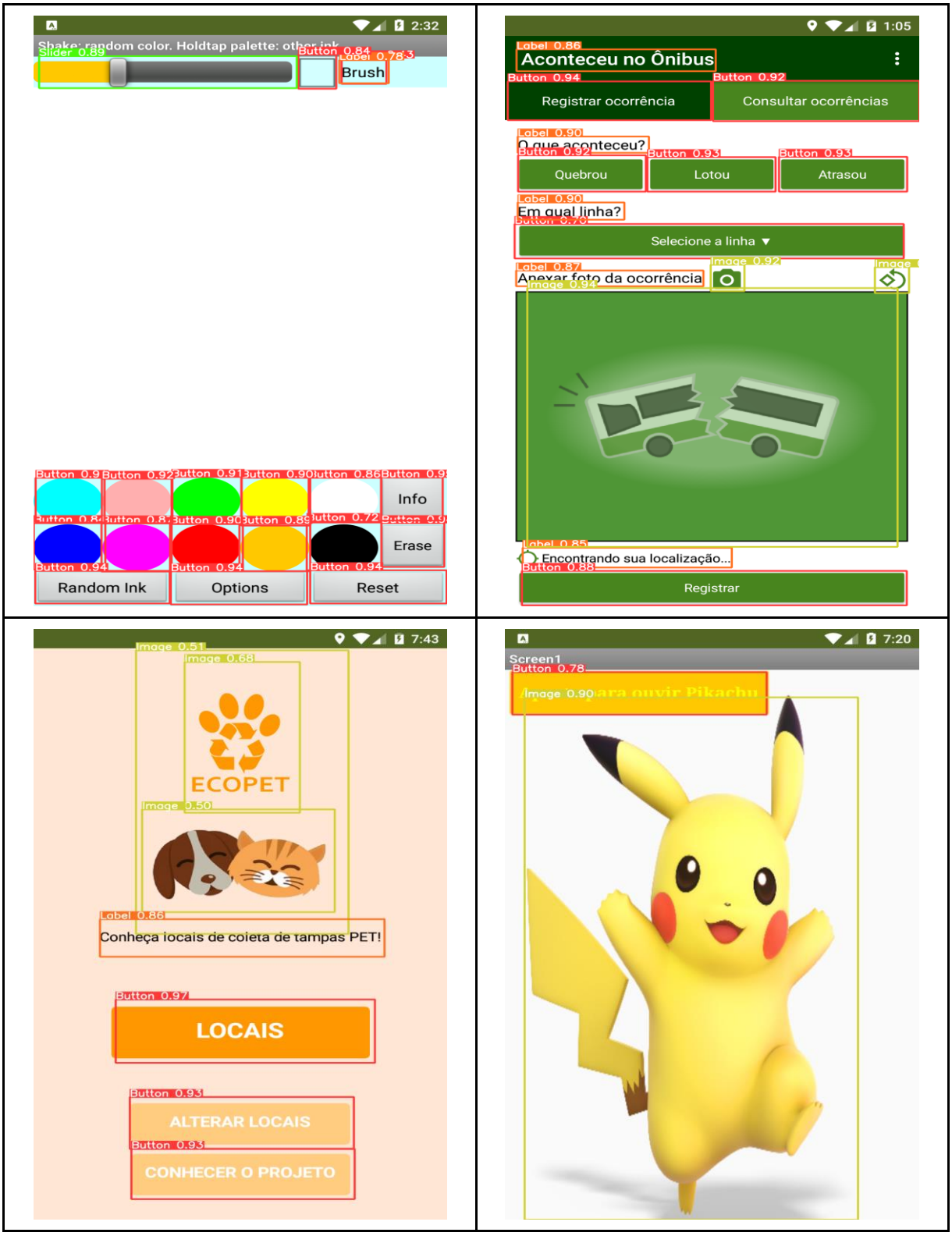
Nota-se que há poucas confusões, como por exemplo entre os elementos Mapa/Botão e Imagem/Botão, possivelmente relacionado ao fato da forma como alguns apps são criados.

Além da análise dos resultados da validação, foi realizado também uma avaliação de desempenho utilizando um conjunto de teste composto de 10 imagens novas que não foram utilizados anteriormente no treinamento/validação.

Os resultados desse teste com a predição utilizando o modelo treinado estão apresentados na figura 26, indicando a detecção dos componentes de forma satisfatória.

Figura 26 – Grau de predição que a rede classifica cada componente de uma tela




2:32
Brush
Info
Erase
Random Ink
Options
Reset
1:05
Aconteceu no Ônibus
Registrar ocorrência
Consultar ocorrências
Quebrou
Lotou
Atrasou
Em qual linha?
Selecione a linha
Anexar foto da ocorrência
Encontrando sua localização...
Registrar
7:43
ECOPET
Conheça locais de coleta de tampas PET!
LOCAIS
ALTERAR LOCAIS
CONHECER O PROJETO
7:20
Screen1

Fonte: dados da pesquisa (2022)

Realizando também a predição para o conjunto de imagens do universo de referência, observou-se que em algumas telas bem diferentes do comum, com

diversos componentes poluindo a tela ou com poucos elementos, o modelo às vezes classifica os componentes de maneira errônea (Figuras 27 e 28).

Figura 27 – Exemplo de tela poluída

Fonte: dados da pesquisa (2022)

Nota-se na Figura 27 a imagem de fundo dificulta a classificação de cada botão (*autre*, *papier*, etc...) e na Figura 28 apenas detecta a tela como uma grande imagem ao invés de um botão. Porém observa-se que nesses casos, muito provavelmente inclusive humanos terão dificuldades de classificar corretamente os elementos e sem maiores informações, p.ex., consultando o código em paralelo será impossível realizar uma classificação correta.

Figura 28 – Exemplo de tela com classificação errônea

Fonte: dados da pesquisa (2022).

Apesar de algumas vezes a rede não classificar corretamente alguns componentes, ainda assim é possível notar que os *bounding boxes* são capturados com precisão boa, também relacionado ao fato de o modelo ter um mAP alto. Dessa forma se considera a versão atual de detecção dos componentes aceitável nesse contexto.

4.3ANÁLISE DO GRAU DE SIMILARIDADE

4.3.1Análise do grau de similaridade de uma tela

Utilizando a informação dos componentes e o layout das telas como entrada calcula-se nesse passo um grau de similaridade das telas de um app em relação ao universo de referência. Para essa análise determina-se primeiro o grau de similaridade de uma tela “nova” criada pelo aluno como resultado de aprendizagem em relação a um universo de referência e ao final um grau de similaridade considerando todas as telas de um app.

Para esse fim foi inicialmente criado um universo de referência com os elementos e layout/estrutura pré-detectada nas telas.

4.3.1.1 Criação do universo de referência

Conforme apresentado na Seção 4.2.1, foram utilizadas 8.067 telas. Para essas 8.067 telas foram pré-detectados os componentes/estrutura de design de interface utilizando o modelo de detecção treinado apresentado na seção 4.2.3. Como resultado criou-se um universo de referência representando as informações de elementos e estrutura para cada um dos 8.067 *screenshots.* Analisando a composição do universo de referência foram levantadas as frequências de cada componente mostrados no Quadro 11.

Quadro 11 - Frequência de ocorrência de componentes no universo de referência

Componente	**Frequência no universo de referência**
Botão	33.197
Mapa	103
Legenda	49.814
Imagem	6.712
CaixaDeTexto	4.782
Deslizador	195
CaixaDeSeleção	2.999
Switch	7
EscolheLista	481

Fonte: o autor.

Nota-se que mesmo em um conjunto de dados com 1.551 apps há componentes que são difíceis de serem encontrados, como Mapa, Deslizador, Switch e EscolheLista.

4.3.1.2 Cálculo do valor de similaridade

Para calcular o valor de similaridade de um novo app (predizendo seus componentes e estrutura usando o modelo de detecção de objetos treinado) em

relação a cada um dos apps no universo de referência, experimentou-se a alternativa proposta por Mao *et al.* (2018). Essa abordagem compara a similaridade entre dois apps utilizando os seguintes dados: tipo do componente, coordenadas x, coordenadas y, altura e largura.

Com essas informações sobre os apps, segue-se um conjunto de fórmulas para calcular as taxas de similaridade entre os componentes entre dois apps:

Similaridade do tipo: Dados dois componentes c1 = {c1.t, c1.p, c1.s} e c2 = {c2.t, c2.p, c2.s}, a similaridade do tipo *s.t* entre c1 e c2 pode ser calculada da seguinte forma (1):

$$s.t = \begin{cases} 0 & c1.t \neq c2.t \\ 1 & c1.t = c2.t \end{cases} \quad (1)$$

Similaridade de posição: Dados dois componentes c1 e c2 com suas posições correspondentes c1.p e c2.p, a similaridade de posição *s.p* pode ser calculada da seguinte forma (2):

$$s.p=(1-||c1.p-c2.p||/||lmax||)\times exp[-(||c1.p-c2.p||/||lmax||)] \quad (2)$$

Onde $||c1.p - c2.p||$ é a distância Euclidiana entre os dois componentes, e ‖lmax‖ é o tamanho da diagonal da tela.

Similaridade de tamanho: Dados dois componentes c1 e c2 com seu tamanho correspondente (c1.sw, c1.sh) e (c2.sw, c2.sh). Primeiro é calculada a similaridade entre a largura s.sw (3) e a altura s.sh (4):

$$s.sw=(1-|c1.sw-c2.sw|/max(c1.sw,c2.sw))\times exp[-(|c1.sw-c2.sw|/max(c1.sw,c2.sw))] \quad (3)$$

$$s.sh=(1-|c1.sh-c2.sh|/max(c1.sh,c2.sh))\times exp[-(|c1.sh-c2.sh|/max(c1.sh,c2.sh))] \quad (4)$$

Depois, é calculada a similaridade entre o tamanho dos componentes da seguinte forma (5):

$$s.s=\sqrt{s.sw}\times s.sh \quad (5)$$

Com os 3 valores acima calculados, é calculada uma taxa de similaridade dos componentes (6).

Similaridade dos componentes:

$$s=s.t\times\sqrt{s.p}\times s.s \quad (6)$$

Para refletir a importância na representação visual, é usado o tamanho para derivar o peso da pontuação de similaridade. A pontuação de similaridade com o peso é calculada da seguinte forma (7):

$$sW=s\times c1.sw\times c1\sum.shc1i\in F1c1i.sw\times c1i.sh \quad (7)$$

Após calculada a taxa de similaridade de cada componente, é feita uma comparação de semelhança entre pares para obter uma matriz de similaridade dos componentes do app novo e das telas do universo de referência. Então, para obter o maior valor de taxa de similaridade entre duas telas é utilizado o algoritmo húngaro de Munkres (MUNKRES, 1957), para tratar a matriz de similaridade como uma matriz de custo e obter a melhor correspondência. Como resultado, é determinado um valor de similaridade de uma tela em relação a todas as telas incluídas no universo de referência.

Para fins de avaliação do desempenho, são utilizadas alternativas como a média e o *threshold:* A média é calculada de forma simples, para cada tela é calculada a sua similaridade com o universo de referência e então é feita a divisão pelo número total de telas do universo de referência. O *threshold* é calculado realizando um ranking das telas/apps mais similares, com base em um critério.

4.3.2 Avaliação do desempenho do cálculo do valor de similaridade de um app

Para avaliar o desempenho dessa abordagem foi definido um conjunto de teste selecionando 10 apps para fazer uma comparação com as 8.067 telas do universo de referência (Figura 26) então foi calculada a similaridade de cada tela desses apps, em seguida foi feita uma média ponderada de todas as telas de um app e criado um *ranking* de similaridade utilizando como alternativas o valor de similaridade calculado como a média e usando um *threshold* (Quadro 12). Como resultado é identificado o valor de similaridade de um app levando em consideração todas as suas telas.

A média é calculada de forma simples, soma-se a taxa de similaridade de 1 tela comparada com as 8.067 telas e divide-se pelo número total de tela do universo de referência. O *threshold* é calculado somando a quantidade total de telas que tem uma taxa de similaridade maior que 50%, então é feito um ranking das 10 telas/apps

que obtiveram mais quantidade de telas/apps similares. Assim, essas medidas são comparadas com um ranking alocado por humanos [0…10], que para o presente trabalho foi realizado pelo autor em conjunto com a orientadora e a coorientadora, com 1 sendo o app que tem mais similaridade com as telas do universo e 10 o app menos similar.

Quadro 12 – Comparação dos rankings de similaridade

App	**Cálculo automático**			**Ranking alocação humanos** **1 (mais similar) a 10 (menos similar)**
	Valor de similaridade média calculada automaticamente com universo de referência	**Ranking baseado na média**	**Ranking Threshold**	
Qarvore	0.310311351996047	1	2	1
Ecopet	0.219849100698706	7	3	3
Pikachu	0.259951133820914	2	1	4
AconteceuNoOnibus	0.251465865284430	3	4	5
ServiceHelper	0.234357990716169	6	9	6
ColaboraEscola	0.154673820099472	10	10	8
4521712606248960	0.237008707053094	5	8	2
SuChef	0.245648470141779	4	5	7
TesteFacil	0.196881145351741	8	6	9
4541200731209728	0.162822863702797	9	7	10

Fonte: o autor

Para avaliar a correlação das alternativas utilizou-se coeficientes de correlação. Foi aplicado o coeficiente de correlação de Spearman (SPEARMAN, 1904). O coeficiente de correlação de Spearman mede a força e a direção da associação entre duas variáveis classificadas, que resulta em um valor entre -1 e 1. Quanto mais uma variável aumenta, a outra também aumenta, então temos uma correlação positiva. Ou quanto mais uma variável aumenta, a outra diminui, resultando

numa correlação negativa. O Quadro 13 apresenta os resultados dos coeficientes de Spearman do conjunto de 10 apps de teste, para obter a correlação entre o *ranking* do julgamento humano com os rankings de média e *threshold* (Quadro 12).

Quadro 13 – Coeficientes de correlação de Spearman

Correlação ranking humano com threshold para abordagem com apps	**Correlação ranking humano com média de similaridade para abordagem com apps**
0.503030303	0.7090909091

Fonte: o autor.

Pode-se notar que as correlações são positivos, o que indica que os valores de média/*threshold* estão indo em direção ao *ranking* humano. Os valores por similaridade atingiram uma correlação forte (acima de 0.70), em ambos os testes tela/app. Já a correlação por *threshold* teve um desempenho abaixo e quase teve uma correlação fraca (0.30). Assim, nota-se também que a correlação avaliando a média de similaridade obteve um valor bem melhor que a correlação por *threshold* (RUMSEY, 2021).

Como o valor de correlação por média de similaridade foi bem melhor, optou-se por utilizar ele para então criar uma nota de originalidade.

4.4 CÁLCULO DA NOTA DE ORIGINALIDADE

Para calcular uma nota de originalidade no contexto de Educação Básica dentro do intervalo [0-10], utiliza-se como entrada o valor de similaridade de um app “novo” em relação ao universo de referência. Levando em consideração que um app pode ter várias telas, calcula-se um valor de similaridade para cada tela do app em relação ao universo de referência individualmente e é calculada uma média ponderada dos valores de similaridade das telas.

Dessa forma a nota 10 representa um aplicativo muito original e a nota 0 a um aplicativo nada original em relação a esqueleto de design de interface).

A transformação do grau de similaridade de um aplicativo em uma nota de originalidade é feita por uma transformação usando uma função. A função foi definida com base numa alocação manual de notas de originalidade por especialistas

utilizando os 10 aplicativos de testes e os valores de similaridade calculados automaticamente (Quadro 14).

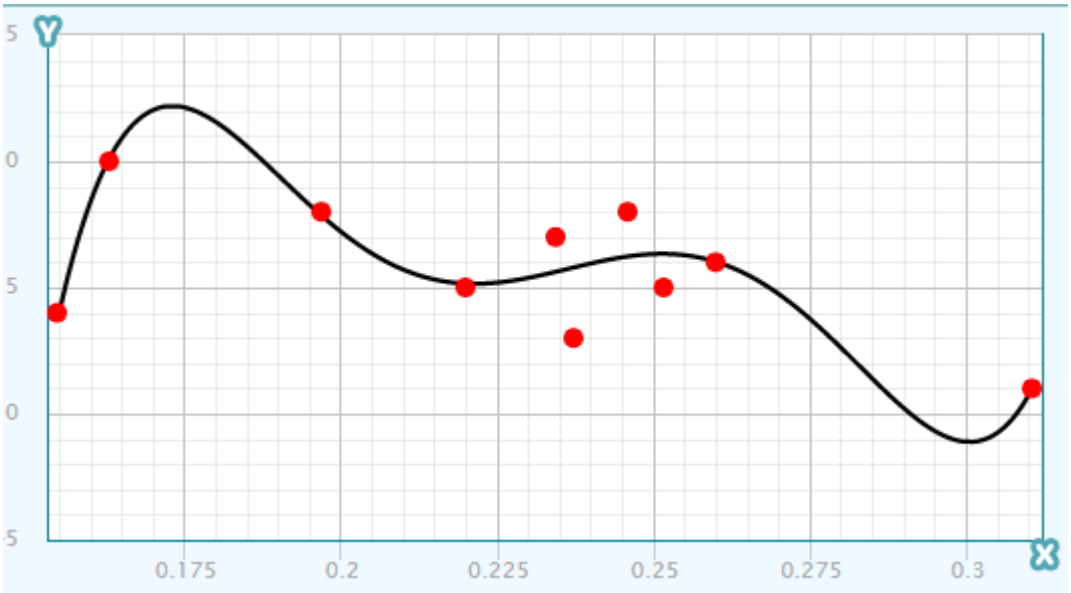
Quadro 14 - Valores de similaridade e notas de originalidade para o conjunto de testes.

App	**Média por app**	**Nota de originalidade alocada por humanos** [1-10] quanto maior mais original
Qarvore	0.310311351996047	1
Ecopet	0.219849100698706	5
Pikachu	0.259951133820914	6
AconteceuNoOnibus	0.251465865284430	5
ServiceHelper	0.234357990716169	7
ColaboraEscola	0.154673820099472	4
4521712606248960	0.237008707053094	3
SuChef	0.245648470141779	8
TesteFacil	0.196881145351741	8
4541200731209728	0.162822863702797	10

Fonte: do autor.

Usou-se a ferramenta *mycurvefit* (https://mycurvefit.com/) para explorar alternativas de funções para analisar quão bem o ajuste de curva modela os dados (Figura 29).

Figura 29- Curva sobre as notas de originalidade alocada por humanos



Fonte: dados da pesquisa (2022).

Foram testadas diversas curvas e a que melhor se aproximou foi a *Polynomial Quintic Regression*. Ainda assim, pode haver um sobreajuste da curva se adaptando ao conjunto de dados e pode ser que a curva não represente tão fielmente o modelo, também pelo fato de que o conjunto de apps selecionados é pequeno.

Dessa forma propõe-se o cálculo da nota de originalidade da seguinte forma: Nota de originalidade = -8437.192 + 188802.4*x - 1663437*x^2 + 7225308*x^3 - 15483720*x^4 + 13102680*x^5. Onde x é a soma do grau de similaridade das telas do app.

4.5 IMPLEMENTAÇÃO DA SOLUÇÃO

A implementação da solução foi realizada utilizando o Jupyter Notebook no Google Colab, seguiu-se a própria documentação/tutorial do YOLOv5 para realizar o treinamento, que é muito facilitado pela versatilidade do YOLOv5. Obtendo as saídas do treinamento da rede, foram escritos scripts em Python para adaptar a entrada

necessária e implementar a comparação entre os app e as telas para obter a similaridade e posteriormente uma nota de originalidade.

Tanto a implementação do Jupyter Notebook quanto o script em Python serão disponibilizados na versão final via GitLab da UFSC (https://codigos.ufsc.br/gqs/dl-originalidade-do-esqueleto-de-design/-/tree/master).

5 CONCLUSÃO

Como resultado do presente trabalho foi sintetizada a fundamentação teórica sobre originalidade, design de interfaces de usuário e *deep learning (O1).* Foi analisado também o estado da arte em relação a análise automática da originalidade de design de design de interfaces de apps (O2), observando que atualmente não existem ainda soluções para automaticamente calcular uma nota de originalidade com base nos *screenshots* de telas dos apps. Com base na fundamentação teórica e os resultados da revisão do estado da arte foi desenvolvida uma abordagem para automaticamente avaliar o esqueleto de design de interface de um app criado com App Inventor. Esse modelo de avaliação contém um universo de referência com 1.551 apps, um procedimento que detecta automaticamente os elementos e estrutura do UI usando DL. A partir dessa informação é utilizada uma medida de similaridade comparando o design de um novo app com os designs dos apps no universo de referência de 1.551 apps e que ao final calcula uma nota de originalidade com base no grau de similaridade. Em relação ao conjunto de testes composto por 10 apps, o modelo desenvolvido apresenta um desempenho aceitável.

O principal benefício do modelo desenvolvido é a possibilidade de automaticamente avaliar essa parte no processo de aprendizagem do aluno possibilitando dar um *feedback* instantâneo como também em cursos online sem instrutor. Espera-se que dessa forma possa contribuir ao desenvolvimento de criatividade de alunos na educação básica no contexto do ensino de computação por meio de aplicativos móveis.

Como trabalhos futuros, sugere-se realizar uma busca mais ampla por componentes que aparecem com frequência menor no App Inventor e um conjunto maior de apps para ter um conjunto de dados e uma confiabilidade maior. Recomenda-se também avaliar outros componentes do design visual além dos descritos no presente trabalho, elementos como cor podem mudar significativamente uma nota de similaridade. Como o presente trabalho faz uma comparação automatizada de telas com telas, uma alternativa pode ser uma comparação automatizada de app (considerando todas as telas do app) com apps.

REFERÊNCIAS

ALVES, G. Detecção de Objetos com YOLO – Uma abordagem moderna. In: Expert Academy. [S.I.], 13 out. 2020. Disponível em: https://iaexpert.academy/2020/10/13/deteccao-de-objetos-com-yolo-uma-abordagem-moderna/. Acesso em: 14 fev. 2022.

ALVES, N. da C., GRESSE VON WANGENHEIM, C., MARTINS-PACHECO, L. H., BORGATTO, A. F. Artefatos computacionais são considerados criativos? In: **Anais do Simpósio Brasileiro de Educação em Computação**, Feira de Santana, Bahia, 2022.

ALVES, N. da C. *et al*. Existe consenso na avaliação da criatividade de resultados da aprendizagem de computação na Educação Básica. In **Proc. of Anais do Simpósio Brasileiro de Educação em Computação**, Jataí, Goiás, 2021.

ALVES, N. da C., GRESSE VON WANGENHEIM, C., HAUCK, J. C. R. A Large-scale Analysis of App Inventor Projects. **XV Conferencia Latinoamericana de Tecnologias de Aprendizaje (LACLO)**, 2020.

ALVES, N. da C.; GRESSE VON WANGENHEIM, C.; ALBERTO, M.; MARTINS-PACHECO, L. H. Uma proposta de Avaliação da Originalidade do Produto no Ensino de Algoritmos de Programação na Educação Básica. *In*: **Anais do Simpósio Brasileiro de Informática na Educação**, Porto Alegre, 2020b.

.

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 VII Congresso Brasileiro de Informática na Educação**. Brasília/DF, 2019.

ACM; IEEE; IEEE Computer Society. Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science. **Association for Computing Machinery.** Nova York, NY, EUA, 518 p., 2013.

BEGHETTO, R. Creativity in the Classroom. In KAUFMAN, J.; STERNBERG, R. (Eds.), **The Cambridge Handbook of Creativity**. Cambridge: Cambridge University Press. p. 447-464, 2010.

BEHRANG, F., REISS, S.P., ORSO, A. GUIfetch: supporting app design and development through GUI search. **Proc. of 5th International Conference on Mobile Software Engineering and Systems,** ACM, New York, USA, p. 236-246, 2018.

BESEMER, S., TREFFINGER, D. J. Analysis of Creative Products: Review and Synthesis. **Journal of Creative Behavior**, v. 15, n. 3, p. 158-178, 1981.

BUSHAEV, V. How do we 'train' neural networks? *In:* Towards Data Science. [*S.l.*], 27 nov. 2017. Disponível em: https://towardsdatascience.com/how-do-we-train-neural-networks-edd985562b73. Acesso em: abril 2021.

CHEN, J. et al. Wireframe-based UI Design Search through Image Autoencoder. **ACM Trans. Softw. Eng. Methodol.**, v. 29, n. 3, jul. 2020.

CHEN, X. *et al.* Recommending software features for mobile applications based on user interface comparison. **Requirements Engineering**, v. 24, n. 4, p. 545-559, 2019.

COOPER, A. *et al***. About face: the essentials of interaction design**. Indianápolis: John Wiley & Sons, Inc., 4. ed., 2014.

CORDEIRO, A. M. *et al*. Revisão sistemática: uma revisão narrativa. **Revista do Colégio Brasileiro de Cirurgiões**, v. 36, n. 6, p. 428-431, dez. 2007.

CSTA. ACM. **CSTA K-12 Computer Science Standards**, 2016. Disponível em: https://k12cs.org/wp-content/uploads/2016/09/K%E2%80%9312-Computer-Science-Framework.pdf. Acesso em: abril 2021.

DEKA, B. et al. Rico: A Mobile App Dataset for Building Data-Driven Design Applications. **Association for Computing Machinery**, New York, NY, USA, p. 845–854, 2017.

GARRETT, J. J. **Elements of user experience, the: user-centered de-sign for the web and beyond**. Pearson Education, 2 ed., 2011.

GE, X. Android GUI search using hand-drawn sketches. **Proc. of the 41st International Conference on Software Engineering: Companion**, IEEE Press, p. 141–143, 2019.

GOOGLE. Material Design. 2021. Disponível em: https://www.material.io/. Acesso em: Março 2021.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. Cambridge, MA: MIT Press. 2017.

GRESSE VON WANGENHEIM, C. *et al.* CodeMaster – Automatic Assessment and Grading of App Inventor and Snap! Programs. **Informatics in Education**, v. 17, n. 1, p. 117-150, 2018a.

GRESSE VON WANGENHEIM, C. *et al*. Do we agree on user interface aesthetics of Android apps? **arXiv**:1812.09049[cs.SE], 2018b.

HAYKIN, S. **Redes Neurais: Princípios e Prática**. Porto Alegre: Bookman Editora, 2 ed., 2007.

HENRIKSEN, D.; MISHRA, P.; MEHTA, R. Novel, Effective, Whole: Toward a NEW Framework for Evaluations of Creative Products. **Journal of Technology and Teacher Education**, v. 23, n. 3, p. 455-478, 2015.

HU, Y. et al. Robust App Clone Detection Based on Similarity of UI Structure, In: **Proc. of IEEE Access**, 8, 77142-77155, 2020.

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: março 2021.

JACKSON, D.N.; MESSICK, S.; MYERS, C.T. Evaluation of Group and Individual Forms of Embedded-Figures Measures of Field-Independence. Educational and Psychological Measurement. v. 24, n. 2, p. 177- 192, jul. 1964.

KAUFMAN, J. C.; BEGHETTO, R. A. In praise of Clark Kent: Creative metacognition and the importance of teaching kids when (not) to be creative. **Roeper Review**, v. 35, n. 3, p. 155–165, 2013.

KAUFMAN, J. C.; BEGHETTO, R. A. Beyond Big and Little: The Four C Model of Creativity. **Review of General Psychology**, v. 13, n. 1, p. 1-12, 2009.

KIERSKI, M. Machine Learning development process - you've got it wrong. In: Mariusz Kierski. [*S.l.*], 3 mar. 2017. Disponível em: https://medium.com/sigmoidal/machine-learning-development-process-youve-got-it-wrong-396270e653f4. Acesso em: dezembro 2020.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. **Nature**, v. 521, p. 436–444, mai. 2015.

LEE, I. *et al.* Computational thinking for Youth in Practice, **ACM Inroads**, v. 2, n.1, p. 32-37, mar. 2011.

MAO, J*., et al*. Robust Detection of Android UI Similarity. School of Electronic and Information Engineering, Beihang University, 2018.

MISHRA, P.; HENRIKSEN, D. A NEW Approach to Defining and Measuring Creativity: Rethinking Technology & Creativity in the 21st Century. **TECHTRENDS**, n. 57, p. 10-13, 2013.

MASSACHUSETTS INSTITUTE OF TECHNOLOGY. **MIT App Inventor**. Boston, 2021. Disponível em: http://appinventor.mit.edu/. Acesso em: fevereiro, 2022.

MUNKRES, J. Algorithms for the Assignment and Transportation Problems. **Journal of the Society for Industrial and Applied Mathematics**, v. 5, n. 1, p.32–38, mar. 1957.

MUSTAFARAJ, E.; TURBAK, F.; SVANBERG, M. Identifying Original Projects in App Inventor. **Florida Artificial Intelligence Research Society Conference**, Massachusetts, USA, maio. 2017. Disponível em: https://www.aaai.org/ocs/index.php/FLAIRS/FLAIRS17/paper/view/15549/15003. Acesso em: fevereiro, 2022

MURTAZIN, E. Apple's Phone: From 1980s' Sketches to iPhone. Part 3. *In:* Mobile Review [*S.l.*] 20 jun. 2010. Disponível em: https://mobile-review.com/articles/2010/iphone-history3-en.shtml. Acesso em: 19 mar. 2021.

NGUYEN, T., VU, H., PHAM. Deep Learning UI Design Patterns of Mobile Apps. *In:* **Proc. of IEEE/ACM 40th International Conference on Software Engineering: New Ideas and Emerging Technologies Results (ICSE-NIER),** p. 65-68, 2018.

NIELSEN, J. **Usability Engineering**. Morgan Kaufmann Publishers; edição revisada, 1994.

O'QUIN, K. Creative product scale: applications in product improvement. In COLEMONT, P. *et al.* (org) **Creativity and Innovation: towards a European Network.** Report of the First European Conference on Creativity and Innovation, 'Network in Action',Delft, The Netherlands, dez. 1987, p. 173- 180.

P21, Partnership for 21st century learning, **P21 Framework Definitions**. 2009. Disponível em: https://files.eric.ed.gov/fulltext/ED519462.pdf. Acesso em: fevereiro, 2022.

P21, Framework for 21st century learning Definitions**, Partnership for 21st century learning**. 2019. Disponível em: http://static.battelleforkids.org/documents/p21/P21_Framework_DefinitionsBFK.pdf. Acesso em: fevereiro, 2022

PATHMIND. A Beginner's Guide to Neural Networks and Deep Learning. 2020. Disponível em: https://wiki.pathmind.com/neural-network . Acesso em: abril 2021.

PETERSEN, K.; VAKKALANKA, S.; KUZNIARZ, L. Guidelines for conducting systematic mapping studies in software engineering: An update. **Information and Software Technology**, v. 64, p. 1-18, 2015.

PETERSEN, K. *et al.* Systematic Mapping Studies in Software Engineering. In: **Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering**, Bari, Italy, p. 68-77, 2008.

PIASECKI, J., WALIGORA, M., DRANSEIKA, V. Google Search as an Additional Source in Systematic Reviews. **Science And Engineering Ethics**, [s. l], v. 24, p. 809-810, 2018.

POLYZOTIS, N. *et al*. Data Management Challenges in Production Machine Learning. **Proceedings of the ACM International Conference on Management of Data**, Chicago, IL, USA, p. 1723 – 1726, maio 2017.

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

RANJAN B.S.C.; SIDDHARTH, L.; CHAKRABARTI, A. A systematic approach to assessing novelty, requirement satisfaction, and creativity. **Artificial Intelligence for Engineering Design, Analysis and Manufacturing,** v. 32, n. 4, p. 390-414, 2018.

RHODES, M. An Analysis of Creativity. **The Phi Delta Kappan**, v. 42, n.7, p. 305-310, abr. 1961.

RITCHIE, G. D. Assessing Creativity. In **Proceedings of AISB Symposium on AI and Creativity in Art and Science Society for the Study of Artificial Intelligence and Simulation of Behaviour.** 2001. Disponível em: http://www.aisb.org.uk/convention/aisb01/abstracts/index.html. Acesso em: fevereiro, 2022.

RUMSEY, D.J. How to Interpret a Correlation Coefficient r. *In:* Dummies, a very brand [*S.l.*], 08 jul 2021. Disponível em: https://www.dummies.com/article/academics-the-arts/math/statistics/how-to-interpret-a-correlation-coefficient-r-169792. Acesso em: 28 fev. 2022

RUNCO, M. A., JAEGER, G. J. The Standard Definition of Creativity. **Creativity Research Journal**, v. 24, n. 1, p. 92-96, 2012.

RUSSEL, S.J.; NORVIG, P. **Artificial Intelligence: A Modern Approach**. New Jersey: Prentice Hall, 3 ed., 2009.

SAMUEL, A.L. Some Studies in Machine Learning Using the Game of Checkers. **IBM Journal of Research and Development,** v. 3, n. 3, p. 210-229, 1959.

SCHLATTER, T., LEVINSON, D. **Visual usability: Principles and practices for designing digital applications**. Morgan Kaufmann, 2013.

SKAGER, R.W.; SCHULTZ, C. B.; KLEIN, S.P. Points of View about Preference as Tools in the Analysis of Creative Products. **Perceptual and Motor Skills**, v. 22, n. 1, p. 83-94, 1966.

SPEARMAN, C. The Proof and Measurement of Association between Two Things. **The American Journal of Psychology,** v. 15, n. 1, p. 72–101, jan. 1904.

SOLAWETZ, J. The Ultimate Guide to Object Detection. *In:* Blog Roboflow. [*S.l.*], 1 fev. 2021. Disponível em: https://blog.roboflow.com/object-detection/. Acesso em: 09 fev. 2022.

SOLECKI, I.S. *et al.* CodeMaster UI design - app inventor: a rubric for the assessment of the interface design of Android apps developed with app inventor. **Association for Computing Machinery**, New York, NY, USA, Article 17, p. 1–10, 2019.

SVANBERG, M. Using feature vector representations to identify similar projects in app inventor. In: **Proc. Of IEEE Blocks and Beyond Workshop**, p. 117–18, 2017.

TURBAK, F*. et al*. Work in progress: Identifying and analyzing original projects in an open-ended blocks programming environment. In: **Proc. of the 23rd Int. Conference on Distributed Multimedia Systems, Visual Languages and Sentient Systems**, p. 115-117, 2017.

ULTRALYTICS. YOLOv5 Documentation, [s.d.]. Disponível em: https://docs.ultralytics.com/. Acesso em: novembro 2021

WALIA, C. A Dynamic Definition of Creativity. **Creativity Research Journal**, v. 31, n. 3, p. 237-247, ago. 2019.

WASSERMAN, A. I. Software engineering issues for mobile application development. **Proc. of Workshop on Future of Software Engineering Research**, Santa Fe, New México, USA, 2010.

XIE, Y.; LIN, T.; XU., H. User Interface Code Retrieval: A Novel Visual-Representation-Aware Approach. *In*: **Proc. of IEEE Access**, 7, 162756-162767, 2019.

YADAV, A., COOPER, S. Fostering Creativity Through Computing**. Communications of the ACM**, v. 60, n. 2, p. 31-33, fev. 2017.

ZOU, Z. *et al*. Object Detection in 20 Years: A Survey, **arXiv:1905.05055v2** [cs.CV]. 2019.

APÊNDICE A

Para realizar o treinamento da rede utilizando o Google Colab foram seguidos os seguintes passos:

Setup

Clone repo, install dependencies and check PyTorch and GPU.

```
[ ] !git clone https://github.com/ultralytics/yolov5  # clone
    %cd yolov5
    %pip install -qr requirements.txt  # install

    from yolov5 import utils
    display = utils.notebook_init()  # checks
```

```
[ ] #apos importar manualmente a pasta train_data.zip para dentro da pasta yolov5, rode este comando para extraí-lo
    %cd /content/yolov5/

    !unzip -q ./train_data.zip -d ./
```

```
[ ] #eh realizado o treinamento da rede passando os paramentros, batch, epochs e os pesos pre-definidos --weights do yolov5
    !python train.py --img 640 --batch 2 --epochs 250 --data custom_data.yaml --weights yolov5s.pt --cache
```

```
[ ] #apenas para salvar todo o treinamento e baixar na minha propria maquina
    !zip -r yolov5.zip /content/yolov5
```

APÊNDICE B

Neste apêndice será apresentado o artigo no formato SBC, referente ao presente projeto.

Modelo de avaliação da originalidade do esqueleto de design de interface de aplicativos android

Leonardo Kreuch

Departamento de Informática e Estatística – Universidade Federal de Santa Catarina (UFSC)
Florianópolis – SC – Brasil

Abstract. *With the advancement of technology in the 21st century, new skills are needed in everyday life and in the job market. Among these skills, creativity stands out, which is the ability to create something new and useful. One of the ways to stimulate creativity is through computing in Basic Education, using block programming environments such as App Inventor. In addition, an important point is the feedback that the student receives from the teacher, this feedback can be automated to both give faster feedback to the student and to save the teacher's time. In this context, this article presents the development of an automated model, using techniques to evaluate the originality in the interface design of applications developed with the App Inventor environment, to be used in the teaching of computing in Basic Education.*

Resumo. *Com o avanço da tecnologia no século XXI, novas habilidades se fazem necessárias na vida cotidiana e no mercado de trabalho. Dentre essas habilidades, destaca-se a criatividade, que é a capacidade de criar algo novoe útil. Uma das formas de estimular a criatividade é por meio da computação na Educação Básica, utilizando ambientes de programação em blocos como o App Inventor. Além disso, um ponto importante é o feedback que o aluno recebe do professor, esse feedback pode ser automatizado para tanto dar um feedback mais rapidamente para o aluno quanto para poupar o tempo do professor. Nesse contexto, este artigo apresenta o desenvolvimento de um modelo automatizado, utilizando técnicas para avaliar a originalidade no design de interface de aplicativos desenvolvidos com o ambiente App Inventor, para ser utilizado no ensino de computação na Educação Básica.*

1. Introdução

A criatividade é considerada uma das principais competências do século XXI, essencial para o sucesso profissional e pessoal [for 21st Century Learning 2015]. Segundo [Runco and Jaeger 2012] esta competência tem uma definição padrão dividida em 2 características que são ressaltadas necessárias para criatividade como a originalidade e a eficácia. A criatividade pode se referir a quatro diferentes pontos de vista: pessoas, processos, imprensa e produtos [Rhodes 1961]. A vertente produtos é a que será abordada neste trabalho. Ela se refere aos resultados tangíveis do processo criativo (ou seja, obras de arte, programas de computador entre outras). Assim, um produto criativo pode ser considerado algo que é original, útil e eficaz [Runco and Jaeger 2012], sendo a originalidade uma das principais características da criatividade. Uma ideia ou produto criativo é considerado original se representar algo novo ou que não existia antes [Runco and Jaeger 2012].

Como a criatividade faz parte das habilidades do século XXI, promover seu ensino desde a Educação Básica é muito importante [for 21st Century Learning 2015]. Geralmente, a criatividade está associada às artes e músicas, porém, ela pode ser ensinada também por meio da computação [da Cruz Alves et al. 2020]. A computação tem o potencial para propiciar aos usuários oportunidades para utilizar suas expressões criativas para resolver problemas, criar artefatos computacionais, e desenvolver novos conhecimentos [Yadav and Cooper 2017].

A computação na Educação Básica é tipicamente ensinada com enfoque no ensino de algoritmos e programação usando metodologias ativas, levando os alunos a criar artefatos computacionais como jogos e apps por meio de ambientes de programação visual baseados em blocos, como o Scratch ou App Inventor. Uma das alternativas consiste em ensinar o desenvolvimento de apps com o App Inventor [Turbak et al. 2017], que suporta tanto a programação da parte funcional quanto o design de interface de usuário, para a criação de apps Android. Assim, os estudantes aprendem tanto conceitos de algoritmos e programação como também de design de interface, que fazem parte do corpo de conhecimento de computação [Lunt et al. 2008]. Já que o design da interface visa maximizar a usabilidade e a experiência do usuário [Cooper et al. 2014] e afeta diretamente a qualidade de uso do aplicativo pelo usuário [Nielsen 1994].

Para ensinar o desenvolvimento de apps na Educação Básica é tipicamente adotado o ciclo "*Use-Modify-Create*" [Lee et al. 2011] guiando a progressão de aprendizagem. No estágio *Use*, os alunos seguem um tutorial criando um artefato computacional predefinido. No estágio *Modify*, os alunos modificam esse artefato, alterando características e criando novas funcionalidades. Progressivamente, os alunos ganham confiança e avançam para o estágio *Create*, no qual eles são incentivados a desenvolverem seus próprios artefatos computacionais com as características que lhes convém, abordando temas de sua escolha. Assim, os alunos desenvolvem a capacidade de gerar ideias e soluções originais e úteis durante as etapas de modificação e principalmente na criação [Walia 2019].

Um exemplo desta progressão é apresentado na (Figura 1), onde há os três passos do ciclo "*Use-Modify-Create*" (UMC) detalhadamente.

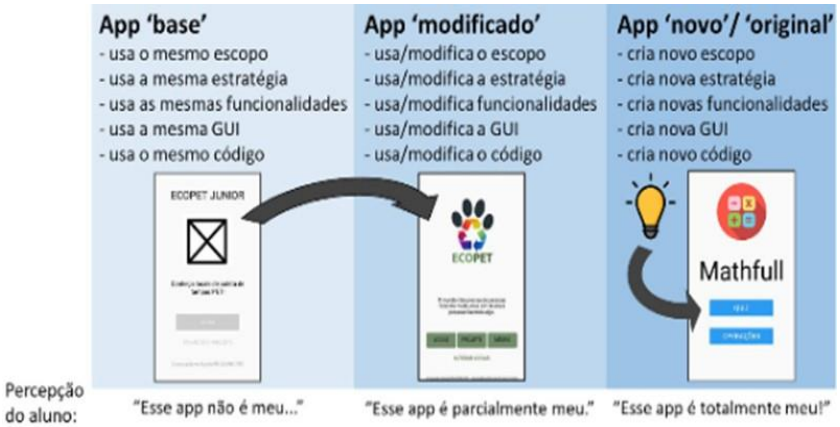


Figure 1. Abordagem UMC no contexto de desenvolvimento de apps

Como parte do processo de ensino-aprendizagem, é importante também a avaliação da criatividade pelo desempenho a partir dos artefatos desenvolvidos pelos alunos. Assim, visa-se avaliar como parte da criatividade a originalidade dos artefatos criados pelos alunos. Como esse processo de avaliação requer um esforço considerável

por parte do avaliador e também se trata de um julgamento humano, pode ser influenciado por aspectos pessoais e valores culturais [von Wangenheim et al. 2018].

Para acompanhar o desenvolvimento da criatividade dos alunos, é fundamental haver uma forma de fornecer um feedback [Mishra and Henriksen 2013], incluindo também um feedback sobre a originalidade do design de interface de usuário criada pelo aluno como resultado da aprendizagem. Entretanto, o processo de avaliação da criatividade é um desafio, pois a criatividade é difícil de medir de forma objetiva com confiabilidade e validade [Henriksen et al. 2015]

Esse processo de avaliação requer um esforço considerável por parte do avaliador e do risco de introduzir um viés se for feito por meio de um julgamento humano, podendo ser influenciado por aspectos pessoais e valores culturais [von Wangenheim et al. 2018]; [da Cruz Alves et al. 2021]. Assim, uma alternativa é automatizar o processo de avaliação da originalidade do design de interface para aumentar a confiabilidade e validade dos resultados da avaliação, além de minimizar o tempo e esforço dos professores da Educação Básica referente a essa tarefa.

Portanto, o objetivo desse artigo é apresentar um modelo de análise da originalidade de esqueleto de design de interface de apps Android desenvolvidos com App Inventor. Com base nos screenshots das telas dos apps são automaticamente detectados os elementos de UI e a sua posição. São adotadas técnicas de deep learning comparando o esqueleto do design de interface de novo app com os esqueletos de design de interface de apps de um universo de referência e identificado um grau de similaridade/diferença. E, a partir desse grau de similaridade/diferença é calculada uma nota de originalidade dentro do contexto da Educação Básica (anos finais do Ensino Fundamental).

2. Originalidade do design de interface de apps

O App Inventor [Mit 2022] é um ambiente de programação visual amplamente utilizado baseado em blocos para criar aplicativos móveis para dispositivos Android. Ele pode ser facilmente acessado e utilizado gratuitamente por meio de um navegador web. Atualmente, o App Inventor tem mais de 800 mil usuários ativos por mês em 195 países pelo mundo [Mit 2022]. O App Inventor torna a programação mais acessível a pessoas com pouco ou nenhum conhecimento em programação, pois a aplicação é desenvolvida por meio de blocos visuais de comandos, sem necessidade de compreender e memorizar sintaxes de linguagens textuais [Love and Strimel 2016].

O desenvolvimento de um aplicativo no App Inventor é dividido em duas partes: Designer e Blocos. A parte de Blocos suporta/consiste na programação lógica dos componentes do aplicativo, por meio de funções, estruturas de seleções, laços, etc. Os componentes da interface são mostrados no canto esquerdo, onde é possível fazer a interação lógica por meio dos blocos apresentados pelo App Inventor. A parte de Designer permite configurar e arrastar os componentes de interface, como botões, imagens, sensores, sons entre outros. Além disso, todos os componentes podem ter sua aparência configurada da maneira que o usuário desejar, alterando propriedades como tamanho e cor.

O App Inventor também possibilita a organização do Layout do aplicativo, permitindo alinhar os componentes da interface seguindo uma estrutura, com objetivo de ajudar os usuários a entenderem a interface.

3. Modelo de avaliação da originalidade de design de interface de aplicativos androids

O modelo recebeu como entrada um conjunto de *screenshots* de telas de um app "novo" criado por um estudante como resultado de processo de aprendizagem com App Inventor e como resultado indica uma nota de originalidade do esqueleto do design de interface desse app em comparação com um universo de referência. O universo de referência foi composto de um conjunto de apps criados com App Inventor em contextos semelhantes e para os quais já foram automaticamente detectados os elementos e o layout das telas. O processo foi dividido nos seguintes passos:

Inicialmente foi selecionado um conjunto de *screenshots* para ser utilizado como dados para treinamento e validação e foi feita a rotulação dos componentes desses *screenshots* manualmente; É feito o treinamento da rede com base nesses *screenshots* já rotulados; É realizada a detecção automática dos elementos (posição, tamanho e tipo do elemento de UI) nos *screenshots* do universo de referência do novo app usando *Deep Learning*; Comparação dos elementos e layout das telas do app novo com o universo de referência identificando um grau de similaridade par a par com todos os apps no universo; Cálculo de uma nota de originalidade no intervalo de [0...10] com base no grau de similaridade.

A Figura 2 exemplifica o fluxo de trabalho idealizado.

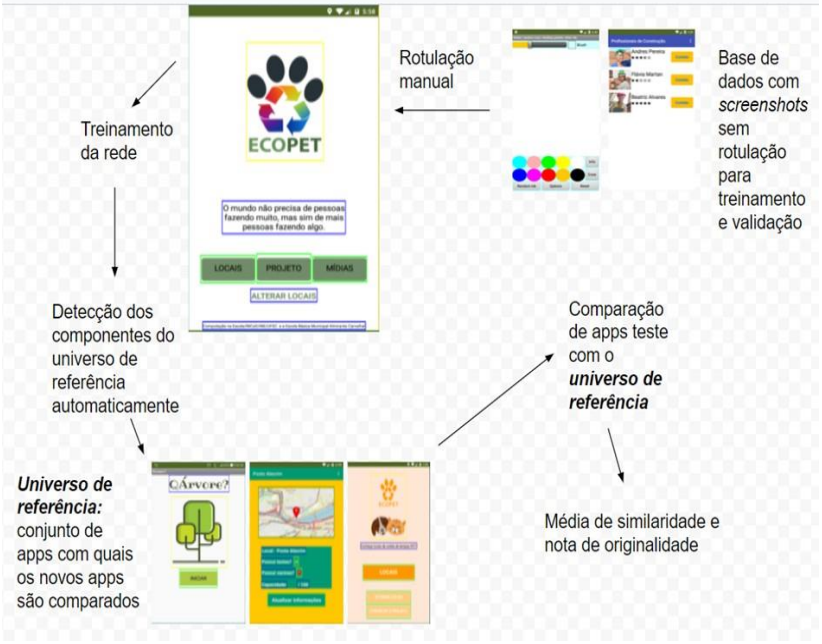


Figure 2. Fluxo de trabalho idealizado

3.1. Preparação do conjunto de dados

Coleta de dados: Levando em consideração o foco do presente trabalho em apps criados com App Inventor, foi criado um conjunto de dados incluindo *screenshots* de apps criados com App Inventor. Esses *screenshots* foram coletados de 1.551 apps selecionados aleatoriamente de um conjunto de mais de 88 mil apps da galeria do App Inventor ou criados no contexto da iniciativa Computação na Escola. A Figura 20 representa alguns exemplos destes *screenshots* selecionados.

No total, foram coletados 8.432 *screenshots* de interface de aplicativos de App Inventor, foram utilizados 292 para o treinamento e 73 para validação. Os outros 8.067

foram utilizados para a criação do universo de referência.

Rotulação dos dados: Para o treinamento do modelo foi adotado a aprendizagem supervisionada, criando manualmente as anotações referentes ao screenshot. As anotações foram criadas utilizando a ferramenta YoloLabel (https://github.com/developer0hye/Yolo_Label). Visualmente quando a anotação de uma imagem é feita por meio do YoloLabel, é apresentado o bounding box conforme mostrado na Figura 21, com os "retângulos" em volta dos componentes capturados, então, essa ferramenta de rotulação gera como saída um arquivo .txt no formato para o modelo Yolo, com os seguintes atributos: class, center_x, center_y, width, height identificando as anotações com as classes de componentes de UI e as posições dos componentes (Figura 3).

Figure 3. Exemplo de *screenshot* e arquivo com o conjunto de labels

3.2. Treinamento do modelo de detecção de componentes

Seleção do Modelo:

Foi utilizado o modelo YOLO (You Only Look Once) de detecção de objetos em imagem. O YOLO é um algoritmo de detecção de objetos que divide imagens em um sistema de grade. Cada célula na grade é responsável por detectar objetos dentro de si. O YOLO é um dos algoritmos de detecção de objetos mais famosos devido à sua velocidade e precisão (ULTRALYTICS, 2021). Para o presente trabalho, foi selecionado o YOLOv5s por ser a versão atual e por motivos tecnológicos, pois os outros modelos ex- igem mais memória e CUDA para treinar o modelo e são muito mais lentos para executar o treinamento.

Treinamento de modelo: O treinamento foi realizado utilizando pesos pré-treinados disponibilizados pelo próprio YOLOv5 e adaptando os arquivos necessários, modificando o número de classes, epochs e batch_size conforme a Figura 4.

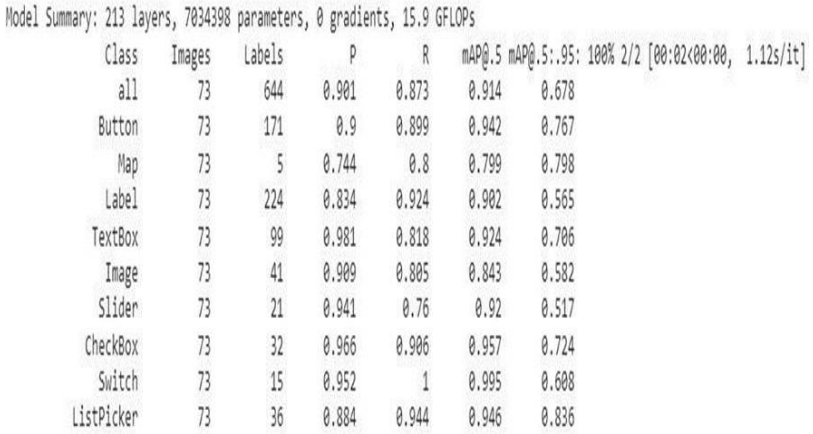
Quantidade de épocas	O treinamento foi iniciado com 300 épocas, porém, como após 175 épocas não foi identificado melhora do desempenho do modelo, a quantidade de épocas foi reduzida para 175 para evitar o overfitting.
Tamanho do *batch*	32
Quantidade total	365 imagens
Divisão do conjunto de dados	O conjunto de dados foi dividido aleatoriamente em um conjunto de treinamento com 292 imagens (80%) e um conjunto de validação com 73 imagens (20%)

Figure 4. Parâmetros utilizados no treinamento do modelo

Na avaliação do treinamento do modelo de detecção de objetos várias medidas

foram analisadas, disponibilizadas pelo próprio YOLOv5 (Figura 5).

Nota-se que o mAP (IOU = 0.5) atinge uma precisão de 91.4% que é excelente. O mAP (IOU 0.5:0.95) atinge uma precisão de 67.8% o que é aceitável pois é calculado com um *threshold* de valores entre 0.50 até 0.95.



Model Summary: 213 layers, 7034398 parameters, 0 gradients, 15.9 GFLOPs

Class	Images	Labels	P	R	mAP@.5	mAP@.5:.95: 100% 2/2 [00:02<00:00, 1.12s/it]
all	73	644	0.901	0.873	0.914	0.678
Button	73	171	0.9	0.899	0.942	0.767
Map	73	5	0.744	0.8	0.799	0.798
Label	73	224	0.834	0.924	0.902	0.565
TextBox	73	99	0.981	0.818	0.924	0.706
Image	73	41	0.909	0.805	0.843	0.582
Slider	73	21	0.941	0.76	0.92	0.517
CheckBox	73	32	0.966	0.906	0.957	0.724
Switch	73	15	0.952	1	0.995	0.608
ListPicker	73	36	0.884	0.944	0.946	0.836

Figure 5. Visão geral das medidas de desempenho mAP

4. Análise do grau de similaridade

Utilizando a informação dos componentes e o layout das telas como entrada calcula-se nesse passo um grau de similaridade das telas de um app em relação ao universo de referência. Para essa análise determina-se primeiro o grau de similaridade de uma tela "nova" criada pelo aluno como resultado de aprendizagem em relação a um universo de referência e ao final um grau de similaridade considerando todas as telas de um app.

4.1. Cálculo do valor de similaridade

Para calcular o valor de similaridade de um novo app (predizendo seus componentes e estrutura usando o modelo de detecção de objetos treinado) em relação a cada um dos apps no universo de referência, experimentou-se a alternativa proposta por [Mao et al. 2018]. Essa abordagem compara a similaridade entre dois apps utilizando os seguintes dados: tipo do componente, coordenadas x, coordenadas y, altura e largura.

5. Avaliação do desempenho do cálculo do valor de similaridade de um app

Para fins de avaliação do desempenho, são utilizadas alternativas como a média e o *thresh*-old: A média é calculada de forma simples, para cada tela é calculada a sua similaridade com o universo de referência e então é feita a divisão pelo número total de telas do universo de referência. O *threshold* é calculado realizando um ranking das telas/apps mais similares, com base em um critério.

Para avaliar o desempenho dessa abordagem foi definido um conjunto de teste selecionando 10 apps para fazer uma comparação com as 8.067 telas do universo de referência e então foi calculada a similaridade de cada tela desses apps, em seguida foi feita uma média ponderada de todas as telas de um app e criado um ranking de similaridade utilizando como alternativas o valor de similaridade calculado como a média e usando um *threshold* (Figura 6). Como resultado e´ identificado o valor de similaridade de um app levando em consideração todas as suas telas.

A média é calculada de forma simples, soma-se a taxa de similaridade de 1 tela

comparada com as 8.067 telas e divide-se pelo número total de tela do universo de referência. O *threshold* é calculado somando a quantidade total de telas que tem uma taxa de similaridade maior que 50%, então é feito um ranking das 10 telas/apps que obtiveram mais quantidade de telas/apps similares. Assim, essas medidas são comparadas com um ranking alocado por humanos [0. . . 10], que para o presente trabalho foi realizado pelo autor em conjunto com a orientadora e a coorientadora, com 10 sendo o app que tem mais similaridade com as telas do universo e 1 o app menos similar.

App	Cálculo automático			Ranking alocação humanos
	Valor de similaridade média calculada automaticamente com universo de referência	Ranking baseado na média	Ranking Threshold	
Qarvore	0.310311351996047	1	2	1
Ecopet	0.219849100698706	7	3	3
Pikachu	0.259951133820914	2	1	4
AconteceuNoOnibus	0.251465865284430	3	4	5
ServiceHelper	0.234357990716169	6	9	6
ColaboraEscola	0.154673820099472	10	10	8
4521712606248960	0.237008707053094	5	8	2
SuChef	0.245648470141779	4	5	7
TesteFacil	0.196881145351741	8	6	9
4541200731209728	0.162822863702797	9	7	10

Figure 6. Comparação dos rankings de similaridade

Para avaliar a correlação das alternativas utilizou-se coeficientes de correlação. Foi aplicado o coeficiente de correlação de Spearman [Spearman 1961]. O coeficiente de correlação de Spearman mede a força e a direção da associação entre duas variáveis classificadas, que resulta em um valor entre -1 e 1. Quanto mais uma variável aumenta, a outra também aumenta, então temos uma correlação positiva. Ou quanto mais uma variável aumenta, a outra diminui, resultando numa correlação negativa. A figura 7 apresenta os resultados dos coeficientes de Spearman do conjunto de 10 apps de teste, para obter a correlação entre o ranking do julgamento humano com os rankings de média e *threshold* (Figura 6).

Correlação ranking humano com threshold para abordagem com apps	Correlação ranking humano com média de similaridade para abordagem com apps
0.503030303	0.7090909091

Figure 7. Coeficientes de correlação de Spearman

Pode-se notar que as correlações são positivas, o que indica que os valores de média/*threshold* estão indo em direção ao ranking humano. Os valores por similaridade atingiram uma correlação forte (acima de 0.70), em ambos os testes tela/app. Já a correlação por *threshold* teve um desempenho abaixo e quase teve uma correlação fraca

(0.30). Assim, nota-se também que a correlação avaliando a média de similaridade obteve um valor bem melhor que a correlação por *threshold* [Rumsey 2016].

Como o valor de correlação por média de similaridade foi bem melhor, optou-se por utilizar ele para então criar uma nota de originalidade.

6. Cálculo da nota de originalidade

Para calcular uma nota de originalidade no contexto de Educação Básica dentro do intervalo [0-10], utiliza-se como entrada o valor de similaridade de um app "novo" em relação ao universo de referência. Levando em consideração que um app pode ter várias telas, calcula-se um valor de similaridade para cada tela do app em relação ao universo de referência individualmente e é calculada uma média ponderada dos valores de similaridade das telas.

Dessa forma a nota 10 representa um aplicativo muito original e a nota 0 a um aplicativo nada original em relação a esqueleto de design de interface).

A transformação do grau de similaridade de um aplicativo em uma nota de originalidade é feita por uma transformação usando uma função. A função foi definida com base numa alocação manual de notas de originalidade por especialistas utilizando os 10 aplicativos de testes e os valores de similaridade calculados automaticamente (Figura 8).

App	**Média por app**	**Nota de originalidade alocada por humanos** [1-10] quanto maior mais original
Qarvore	0.310311351996047	1
Ecopet	0.219849100698706	5
Pikachu	0.259951133820914	6
AconteceuNoOnibus	0.251465865284430	5
ServiceHelper	0.234357990716169	7
ColaboraEscola	0.154673820099472	4
4521712606248960	0.237008707053094	3
SuChef	0.245648470141779	8
TesteFacil	0.196881145351741	8
4541200731209728	0.162822863702797	10

Figure 8. Valores de similaridade e notas de originalidade para o conjunto de testes

Usou-se a ferramenta *mycurvefit* para explorar alternativas de funções para analisar quão bem o ajuste de curva modela os dados (Figura 9).

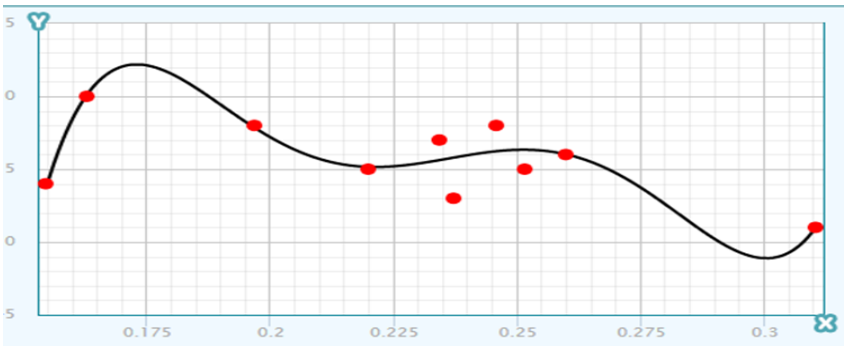


Figure 9. Curva sobre as notas de originalidade alocada por humanos

Foram testadas diversas curvas e a que melhor se aproximou foi a *Polynomial Quintic Regression*. Ainda assim, pode haver um sobreajuste da curva se adaptando ao conjunto de dados e pode ser que a curva não represente tão fielmente o modelo.

Dessa forma propõe-se o cálculo da nota de originalidade da seguinte forma: Nota de originalidade = -8437.192 + 188802.4*x - 1663437*x$\hat{2}$ + 7225308*x$\hat{3}$ - 15483720*x$\hat{4}$ + 13102680*x$\hat{5}$. Onde x é a soma do grau de similaridade das telas do app.

7. Conclusão

Como resultado do presente trabalho foi realizado um modelo de avaliação automática da originalidade de apps feitos no App Inventor. O principal benefício do modelo desenvolvido é a possibilidade de automaticamente avaliar essa parte no processo de aprendizagem do aluno possibilitando dar um *feedback* instantâneo como também em cursos online sem instrutor. Espera-se que dessa forma possa contribuir ao desenvolvimento de criatividade de alunos na educação básica no contexto do ensino de computação por meio de aplicativos móveis.

Como trabalhos futuros, sugere-se realizar uma busca mais ampla por componentes que aparecem com frequência menor no App Inventor e um conjunto maior de apps para ter um conjunto de dados e uma confiabilidade maior. Recomenda-se também avaliar outros componentes do design visual além dos descritos no presente trabalho, elementos como cor podem mudar significativamente uma nota de similaridade. Como o presente trabalho faz uma comparação automatizada de telas com telas, uma alternativa pode ser uma comparação automatizada de app (considerando todas as telas do app) com apps.

References

Cooper, A., Reimann, R., Cronin, D., and Noessel, C. (2014). *About face: the essentials of interaction design*. John Wiley & Sons.

da Cruz Alves, N., von Wangenheim, C., Alberto, M., and Martins-Pacheco, L. (2020). Uma proposta de avaliação da originalidade do produto no ensino de algoritmos e programação na educação básica. In *Anais do XXXI Simpósio Brasileiro de Informática na Educação*, pages 41–50, Porto Alegre, RS, Brasil. SBC.

da Cruz Alves, N., von Wangenheim, C., Martins-Pacheco, L., and Borgatto, A. F. (2021). Existem concordância e confiabilidade na avaliação da criatividade de resultados tangíveis da aprendizagem de computação na educação básica? In *Anais do Simpósio Brasileiro de Educação em Computação*, pages 12–22, Porto Alegre, RS, Brasil. SBC.

for 21st Century Learning, P. P. (2015). P21 framework definitions. *The Partnership for 21st century learning*, pages 1–9.

Henriksen, D., Mishra, P., and Mehta, R. (2015). Novel, effective, whole: Toward a new framework for evaluations of creative products. *Journal of Technology and Teacher Education*, 23(3):455–478.

Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., Malyn-Smith, J., and Werner, L. (2011). Computational thinking for youth in practice. *Acm Inroads*, 2(1):32–37.

Love, T. S. and Strimel, G. J. (2016). Computer science and technology and engineering education: A content analysis of standards and curricular resources. *Journal of Technology Studies*, 42(2):76–89.

Lunt, B., Ekstrom, J., Gorka, S., Hislop, G., Kamali, R., Lawson, E., LeBlanc, R., Miller, J., and Reichgelt, H. (2008). *Curriculum guidelines for undergraduate degree programs in information technology*. ACM.

Mao, J., Bian, J., Ma, H., Jia, Y., Liang, Z., and Jiang, X. (2018). Robust detection of android ui similarity. In *2018 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE.

Mishra, P. and Henriksen, D. (2013). A new approach to defining and measuring creativity: Rethinking technology & creativity in the 21st century. *TechTrends*, 57(5):10.

Mit (2022). App inventor. https://appinventor.mit.edu/. acessado em 01/02/2022.

Nielsen, J. (1994). *Usability engineering*. Morgan Kaufmann.

Rhodes, M. (1961). An analysis of creativity. *The Phi delta kappan*, 42(7):305–310.

Rumsey, D. J. (2016). How to interpret a correlation coefficient r. *Statistics For Dummies*.

Runco, M. A. and Jaeger, G. J. (2012). The standard definition of creativity. *Creativity research journal*, 24(1):92–96.

Spearman, C. (1961). The proof and measurement of association between two things.

Turbak, F., Mustafaraj, E., Svanberg, M., and Dawson, M. (2017). Work in progress: Identifying and analyzing original projects in an open-ended blocks programming environment. In *Proceedings of the The 23rd International DMS Conference on Visual Languages and Sentient Systems (DMSVLSS 2017)*.

von Wangenheim, C. G., Porto, J. V. A., Hauck, J. C., and Borgatto, A. F. (2018). Do we agree on user interface aesthetics of android apps? *arXiv preprint arXiv:1812.09049*.

Walia, C. (2019). A dynamic definition of creativity. *Creativity Research Journal*, 31(3):237–247.

Yadav, A. and Cooper, S. (2017). Fostering creativity through computing. *Communications of the ACM*, 60(2):31–33.