Segurança Android com *Touch Screen* aprimorado com IA.

Valter Iversen Fernandes de Souza.

Orientador: Prof. Walison Joel Barberá Alves.

Curso de Sistemas de Informação – Centro Universitário UNIFAFIBE  
 Bebedouro –SP – Brasil

{sisunifafibe}@unifafibe.com.br

**RESUMO:** Com a popularização dos dispositivos móveis, veio consigo a utilização dos mesmos como ferramentas de trabalho. Em certos ambientes e em uma porção de pessoas distintas, existem dificuldades quanto a utilização das ferramentas modernas de acesso a dispositivo móveis. Levantamos a questão quanto a dificuldade de utilização e a má viabilidade do uso das mesmas para pessoas que pertencem a estes ambientes ou que possuem dificuldades individuais voltadas a ruídos nos sensores responsáveis pela leitura dos padrões para acesso, viabilidade no baixo nível de segurança e/ou o alto preço de dispositivos com sensores mais avançados. Assim, satisfazendo a necessidade de uma ferramenta que intermeie tais obstáculos, trazendo uma aplicação voltada a essas pessoas.

**Palavras-chave:** Segurança. Machine Learning. Aplicações Mobile. Multilayer Perceptron.

# 1. INTRODUÇÃO

A tecnologia está cada vez mais acessível, a ponto de ser normal que até crianças com menos de cinco anos de idade possuírem seus próprios aparelhos celulares ou tablets, a informação está cada vez sendo mais utilizada e processada como mecanismo de disputa no mercado, não é de se espantar que a maioria das empresas dependem de dispositivos móveis para lutar contra a concorrência de mercado.

Com esse crescimento de utilização dos mesmos em ambientes empresariais, vem também a necessidade de privatizar os dados que o dispositivo contém sobre a empresa, pois tais dados, quando bem processados e analisados se tornam o bem mais valioso de uma empresa; é essencial que haja, além das políticas, protocolos de segurança e criptografia, a existência de uma ferramenta de autenticação de usuário para acesso das ferramentas e dados presentes no dispositivo móvel.

“De fato, a maior parte das fraudes eletrônicas não ocorrem devido a falhas nos protocolos de segurança, mas sim devido ao descuido dos usuários que de alguma forma permitem que as suas senhas sejam descobertas por pessoas mal-intencionadas.” (HEINEN, 2002, p.13)

Existem atualmente diversas formas de autenticar usuários, envolvendo dados biométricos, alfanuméricos, sequência de caracteres, desenhos (*PIN*) e muitos outros. Porem existem certos ambientes de trabalhos ou pessoas com características individuais que possuem dificuldades tanto na utilização da ferramenta quanto a quantidade de segurança provida por ela.

Um padrão sequencial alfanumérico ou de caracteres são técnicas que não são viáveis em ambiente empresariais devido à baixa quantidade de segurança provida pelos mesmos, pois, qualquer pessoa que descubra sua sequência pode acessar seus dados; tecnologias biométricas como a leitura de digital tem, além da necessidade de um aparelho de alto custo, o empecilho para aqueles que perderam suas digitais em trabalhos manuais, manuseamento de produtos químicos e outros tipos de ambientes de trabalho que causam as perdas das digitais, fazendo com que os usuário não consigam utilizar de forma correta o sensor, outras tecnologias biométricas como o reconhecimento facial ou de retina necessitam de aparelhos ainda mais caros e não funcionam corretamente em ambientes com pouca luz como em ambientes de trabalhos noturnos ou pessoas que utilizam equipamentos que dificultam ou impedem a coleta de dados dessas ferramentas, como capacetes, óculos, máscaras e outros.

Surge-se então a necessidade de uma ferramenta voltada para esses indivíduos específicos; uma ferramenta da qual é proposta aqui. Utilizando *Touch Screen*, o usuário pode realizar padrões figurativos como: desenhos, pontos específicos em determinadas localidades e até mesmo assinaturas; quaisquer padrões que usuário desejar. Trazendo uma ferramenta barata e a possibilidade de um padrão complexo e o fato de que, mesmo que outras pessoas descubram a sua assinatura, terão dificuldades em executar com os mesmos detalhes nas características utilizadas. A ferramenta proposta visa ser simples e solucionar os problemas encontrados por esses usuários.

# 2. REFERÊNCIA BIBLIOGRÁFICA

## **2.1. Redes neurais**

A ferramenta chave utilizada nesse projeto é a inteligência artificial, as redes neurais são as mais famosas e mais utilizada atualmente em ambientes que necessitam de *machine learning.* Atualmente no mercado, é difícil não encontrar um dispositivo que tenha ou dependa de no mínimo um agente inteligente.

“Atualmente, a IA abrange uma enorme variedade de subcampos, desde áreas de uso geral, como aprendizado e percepção, até tarefas específicas como jogos de xadrez, demonstração de teoremas matemáticos, criação de poesia e diagnóstico de doenças.” (RUSSEL; NORVIG, 2004. p. 4)

Uma rede neural, uma subárea da aprendizagem de máquina, é composta de neurônios em camadas que contêm operações matemáticas para processar dados. Ela tem o objetivo de imitar o funcionamento do cérebro humano, a mesma traz uma boa semelhança e acima de tudo, é extremamente eficaz.

Em visão teórica, uma rede neural é composta por neurônios que são dispostos em camadas que se ativam e enviam sinais para outros neurônios conforme o sinal de outros neurônios enviados para a mesma. A mesma pode ser treinada para que haja ajustes nas condições de ativações de cada neurônio. Como os neurônios são compostos de operações matemáticas, o que podemos realizar é ajustes nos pesos das entradas (valores que alteram os valores originais de entrada), o que resultaria em ativações conforme o resultado esperado.

Será criado neste projeto uma rede neural com aprendizado supervisionado. Para realiza os ajustes nos pesos, precisamos de dados de treinamento que contenham os dados e o resultado esperado, para que a rede neural realize reajustes (a proporção do reajuste é definida pelo programador), dessa forma, a rede neural se adapta conforme os dados providos, para que ela tome decisões baseadas nos dados anteriores.

## **2.2. Padrões**

Para desenvolver uma ferramenta para gerir acesso ao dispositivo de forma segura com *Touch Screen*, é viável a utilização de uma inteligência artificial como entrada de padrões que possam ser reconhecidos como características pessoais do usuário. Tais padrões são analisados aqui.

Segundo Hall (apud HEINEN, 2002, p. 24) reconhecimento de padrões consiste em perceber ou identificar um elemento a partir de uma experiência passada ou conhecimento, ou por algum detalhe deste elemento.

Todos os padrões que serão usados aqui poderão ter variações caso o usuário esteja emocionalmente instável. O que nos traz a mais um pequeno detalhe da segurança. Estudos apontam que estados emocionais estão diretamente ligados a padrões comportamentais; atualmente existem sistemas e técnicas muito usadas para reconhecimento de alteração emocional baseados em análises comportamentais levando em consideração a alterações de hormônios cerebrais, fazendo com que possa se analisar e tomar decisões baseado nestes dados.

Caso haja uma tentativa de acesso indevida ao dispositivo, o indivíduo irá apresentar alterações hormonais no cérebro devido a tensão emocional, levando a alteração comportamental, fazendo com que ele tenha dificuldade na execução planejada da cópia do padrão; um ponto deste fator que pode trazer implicações é que caso o usuário verdadeiro esteja enfrentando uma situação de grande tensão emocional, há grandes chances de seu acesso ser recusado.

Devido ao fator anterior e a vários outros, é indicado que o usuário sempre execute os seus padrões sempre com a mesma mão e na mesma posição de manuseio; fazendo desta forma, unificando a forma de execução, trazemos uma maior precisão de treinamento para a inteligência artificial, garantindo a maior segurança contra falsos usuários e menor chance de acessos verdadeiros negados.

Porem os maiores fatores responsáveis pela segurança são as entradas e a precisão que uma rede neural pode trazer realizando treinamento com essas entradas, trazendo uma maior quantidade de permissões de acesso verdadeiras concedidas e acesso indevidas/falsas negadas.

## **2.2.1. Figura**

A mais relevante e nosso maior foco é a realizações de padrões em forma de figura. Este padrão é utilizado como base em vários outros próximos que são descritos aqui.

A figura da a possibilidade para usuário realizar qualquer forma de desenho/figura, podendo ser por exemplo: desenhos abstratos, desenhos de objetos, escrita de nomes, números, símbolos, assinatura ou até mesmo pontos específicos em partes específicas da tela *Touch Screen*. Sendo que será indicado o uso de uma assinatura, para evitar o esquecimento da mesma, ser um padrão de imagem difícil de ser memorizado e replicado e de pouco acesso para as pessoas.

Além de padrões mais simples gerados pela própria biblioteca Android, utilizaremos visão computacional para processar padrões mais voltados para a figura em si.

“A visão computacional procura modelar e replicar a visão humana usando software e hardware. O processamento de imagens tem uma imagem como parâmetro de entrada e um conjunto de valores numéricos como saída (objetos complexos, vetores, matrizes, etc).” (Rocha, 2018)

## **2.2.2. Velocidade total de execução de figura.**

A velocidade total de realização do padrão é usada para identificação do usuário. A mesma será obtida com base no tempo de realização, desde o começo ao final. A mesma, assim como a pressão varia bastante de acordo com o os hábitos e/ou fisionomia do usuário. A velocidade também é importante pois traz consigo características que expõem a individualidade do usuário devido a seus hábitos, além de que, pessoas que tentam realizar a tentativa de fraude, necessitam de tempo para realização da assinatura/desenho de forma eficaz. Este padrão foi usado por Heinen (2002).

## **2.2.3 Quantidade de levantamento dos dedos**

É levado em conta também a quantidade de vezes que o usuário levantou seu dedo durante a execução da figura. Trazendo uma característica importante da forma como foi realizado o padrão de figura. Este padrão foi usado por Heinen (2002).

## **2.2.4 Tempo de execução durante os períodos de levantamento dos dedos**

Tem base no tempo intervalar entre os levantamentos dos dedos da tela *Touch* *Screen*. Nos dando características de como foi realizada a figura. Podendo-se por exemplo: levantar a importância dos traços finais de assinaturas que específicos usuários utilizam em suas assinaturas.

## **2.2.5 Amostragem da trajetória**

Será uma forma de saber qual percurso foi feito durante a assinatura. Será utilizado levando em consideração a distância entre pontos iniciais e pontos finais intervalares entre os levantamentos do dedo. Este padrão foi usado por Heinen(2002).

## **2.2.6 Máximos horizontais e verticais**

Para uma boa construção dos parâmetros de dimensão da figura, inicializaremos utilizando o tamanho horizontal e vertical após a mesma ser processada. Este padrão foi usado por Pansare e Bhatia (2012).

## **2.2.7 Densidade da assinatura**

A assinatura passa por um processo onde ela será transformada em uma representação de seus pixels em um conjunto de números inteiros. Onde o mesmo armazena a densidade da cor em RGB. Desta forma podemos repartir a imagem em quatro áreas no sentido horizontal para assim termos dados referentes a distribuição da assinatura, contando a quantidade de pixels pintados (preenchidos).

## **2.3. Trabalhos relacionados**

No projeto de Heinen (2002) é utilizado uma rede neural artificial *Cascade-Correlation* que tem seu formato em forma de cascata para reajustes automáticos na camada oculta. E como simulador de redes neurais, foi escolhido o Neuzim que implementa o modelo atual.

Gideon, Kandulna, Kujur, Diana e Raimond (2018) desenvolveram um projeto em que é utilizada uma rede neural convolucional, para identificações de assinaturas. Uma rede neural muito voltada para visão computacional, existem sistemas que são capazes de identificar e classificar tumores cerebrais, identificar objetos, reconhecer faces e outros.

Com uma abordagem semelhante a de Heinen (2002) os cientistas Pansare e Bhatia (2012) utilizam redes neurais *backpropagation* para identificação de assinaturas forjadas e reais. Porem utilizando abordagens de padrões mais complexos em questão de matemática e álgebra.

Há também o interessante projeto para cajueiros africanos de Darwaish, Moradian, Rahmani e Knauer que foi desenvolvido com objetivo de aumentar a segurança de transferências entre agropecuaristas africanos, os mesmos tinham dificuldades relacionadas a segurança e veracidade de transferências. Antigamente para cada cajueiro era dado um *QRCode* impresso em papel, no qual podia danificar-se facilmente ou ocorrer perda. Contudo, eles projetaram uma leitura biométrica off-line para identificação dos cajueiros

# 3. METODOLOGIA

Descrevemos aqui os métodos que foram realizados para a construção do projeto. O mesmo está separado em tópicos para melhor entendimento e descrição da metodologia.

**3.1. Plataformas**

O projeto basicamente será separado em 2 partes, um projeto para treinamento da rede neural e seus pesos viabilizando desempenho e tempo, pois se fossemos colocar o projeto com uma escala de desempenho necessária deste tamanho em dispositivos móveis, não seria viável, pois demoraria horas para treinamento da rede neural, não haveria disponibilidade do mesmo peso para outros dispositivos e dificultaria em manter os pesos sempre atualizados e treinados.

**3.1.1. Android Studio**

Para o desenvolvimento do aplicativo nativo, será usado o Android Studio. Ele é uma IDE (Ambiente de Desenvolvimento Integrado) para desenvolvimento para a plataforma Android. É a ferramenta atual mais utilizada pelos desenvolvedores desta plataforma.

A ferramenta foi escolhida por ter disponibilização gratuita, pelo tamanho da comunidade de usuários e pelas atualizações constantes.

**3.1.2. Visual Studio**

Uma das IDEs mais famosas, também tem suporte para a linguagem C++*.* A mesma nos possibilita escrever, depurar e compilar. Ela foi escolhida pelos mesmos fatores, pela sua disponibilização de versão gratuita e pelo tamanho da comunidade de usuários.

**3.1.3. GitHub**

Para gerir e garantir a qualidade, veracidade e segurança da aplicação e dos seus dados, utilizaremos o gestor de versões GitHub. O mesmo é uma ferramenta moderna que permite gerir todo tipo de documentação e codificação, nosso foco será criar repositórios para controlar versões para os pesos gerados pelo treinamento da rede neural, para o aplicativo para dispositivos móveis e para o software de ambiente desktop. Dessa forma garantiremos um maior controle sobre as ferramentas desenvolvidas.

**4. Reconhecimeto de assinatura**

**4.1. Diagrama de funcionamento**

O aplicativo recebe as suas assinaturas para o cadastro e é enviado para análise, onde sua assinatura é utilizada em treinamentos de uma rede neural. Depois de um curto período de tempo seu cadastro é terminado e você pode validar sua assinatura. Vários usuários podem ser cadastrados.

****

Figura 1. Funcionamento do sistema.

**4.2. Figura**

A figura é a imagem obtida no *Touch* *Screen*, uma figura a desejo do usuário. A figura pode ser obtida com a ferramenta *Android Canvas Drawing*, onde a mesma realiza desenhos na tela baseado nos pixels tocados. A biblioteca também permite realizações de figuras geométricas, porem neste projeto é apenas utilizado caminhos (desenhos simples, sem especificações geométricas). A figura sofre apenas um procedimento de pré-processamento de imagens para melhor aproveitamento de desempenho, características e outros fatores extremamente relevantes.

**4.2.1. Ajuste resolução**

Como a figura não necessita de tantos detalhes, ela sofre um corte de resolução, reduzindo a qualidade da imagem, porem trazendo benefícios em desempenho para o processamento da imagem e de armazenamento devido a redução do tamanho da imagem.

**4.3. Padrões**

Aqui serão descritos a metodologia de obtenção dos padrões.

**4.3.1. Velocidade de realização total**

Essa característica é obtida com base no tempo desde o início da realização até o final (confirmação do término). Este padrão é coletado com bibliotecas simples do *Java* onde é armazenado a hora atual (em forma de milissegundos) no início da realização da figura, depois é subtraído com a hora atual (também em milissegundos) trazendo a diferença em milissegundos entre o início e término.

**4.3.2. Quantidade de levantamento dos dedos**

É levado em conta a quantidade de vezes que o usuário levantou o dedo (retirou da tela *Touch Screen*) durante a realização do padrão. Este padrão também será obtido através da própria biblioteca *Java* da plataforma Android.

## **4.3.3. Tempo de execução durante os períodos de levantamento dos dedos**

Uma metodologia que une os dois anteriores. O tempo é obtido conforme o tempo intervalar entre os levantamentos de dedos, utilizando o mesmo método de subtração da hora atual de início e término do movimento.

## **4.3.4. Amostragem da trajetória**

A cada levantamento dedo, é obtido a coordenadas em pixel X (largura) e Y (altura), da entrada e da saída do movimento, realizando a soma entre todas as amostras até o final da execução.

## **4.3.5. Máximos horizontais e verticais**

É obtida com base no tamanho horizontal e vertical preenchido da figura, a mesma pode ser obtida após o ajuste de expansão onde é retirada a parte branca desnecessária.

**4.3.5.1. Ajuste de expansão**

A figura é reajustada conforme foi utilizada a tela, caso haja partes da figura que não foi utilizada. O ajuste é realizado verificando quais pixels foram preenchidos nos limites de X e Y, armazenando esses limites da esquerda para direita, direita para esquerda, cima para baixo e de baixo para cima.

A imagem tem esse dado retirado após seu ajuste de resolução.

Trazendo uma imagem sem espaço em branco desnecessário.

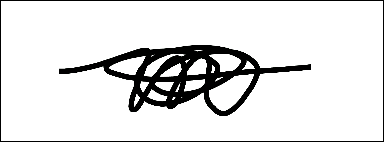


Figura 2. Assinatura sem ajuste.



Figura 3. Assinatura com ajuste.

## **4.3.6. Densidade da assinatura**

Para iniciar este dado, é preciso converter as figuras em um *bitmap* para podermos extrair os valores de cada *pixel* do *bitmap*, valores representados em números inteiros que são retirados de propriedades RGB. A figura é repartida em quatro áreas na horizontal, obtendo dessa forma a distribuição que ocorreu durante a assinatura na imagem. O método para transformar um *bitmap* em um *Array* verifica a propriedade RGB do pixel, e caso não seja branco, o algoritmo realiza uma soma na contagem de pixels pretos por área. Como a imagem é em preto e branco, os dados são de representação mais simples: quanto mais perto de 255 (maior valor possível) mais branco é o pixel, quanto mais perto de 0 (menor valor possível) mais preto é o pixel, ou seja, o pixel abaixo de 255 é um pixel preenchido.

Os valores variam entre 0 e 255 devidamente ao pré-processamento que ajusta a resolução da imagem, fazendo com que pixels brancos e pretos se unam e se tornem pixels acinzentados.

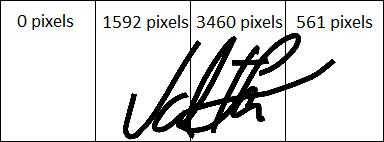


Figura 4. Coleta da densidade da assinatura.

**4.4. Rede neural artificial**

Com estes padrões, é treinada uma *Perceptron Multilayer Backpropagation* com aprendizado supervisionado que define se o acesso é permitido ou não no dispositivo móvel. A mesma é treinada dentro de um ambiente desktop para sempre ocorrer atualizações dos pesos, disponibilização dos pesos atualizados e economizar desempenho, pois, caso o treinamento seja feito no celular, todo dispositivo que tiver o aplicativo instalado, deveria baixar o *Dataset* que contém um imenso número de dados e realizar o treinamento que demoraria horas.

Por motivos de desempenho, o processo de verificação dos dados da assinatura é enviado a uma classe nativa *C++* dentro do código *Java* no dispositivo Android onde contém a rede neural que processará os dados e classificará a assinatura.

A forma de comunicação entre a classe *Java* e *C++* ocorre por meio do padrão de desenvolvimento *JNI (Java Native Interface)* que permite realizar ou receber chamadas de aplicações nativas. O framework é um pouco complexo devidamente a certas regras de declarações e conversões de dados do *Java* para dados nativos.

Mas antes de tudo isso, temos o *NDK (Native Development Kit).*

“NDK é um conjunto de ferramentas que permite usar códigos C e C++ no Android além de fornecer bibliotecas de plataforma que podem ser usadas para gerenciar atividades nativas e acessar componentes físicos do dispositivo, como sensores e entrada de toque” (Android Developers, 2019)

No ambiente desktop, os dados coletados pelo dispositivo Android são tratados e utilizados na rede neural para treinamento de realização de testes. Depois de treinada a rede, os pesos atualizados são disponibilizados para os dispositivos manualmente.

Uma *Perceptron Multilayer* é uma rede neural com várias camadas de neurônios, tendo essas camadas vários neurônios e esses neurônios recebem várias entradas, sendo a primeira camada, a camada de entrada que recebe os dados extraídos da assinatura, ou seja, os padrões descritos anteriormente. Após receber esses valores, a primeira camada distribui seus valores de saída para os outros neurônios da próxima camada, a chamada camada oculta, onde os neurônios recebem suas entradas **x** e atribuem os pesos **w** e a constante bias **b** que também possui seu peso, os pesos são ajustados durante a fase de treinamento. O neurônio utiliza uma função de ativação que tem como entrada a somatória das entradas multiplicados pelos seus pesos e valida se o neurônio vai ser ativado ou não, definindo qual vai ser a entrada para o próximo neurônio da próxima camada.

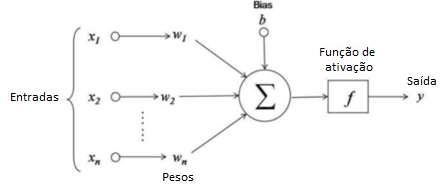


Figura 5. Exemplo de um neurônio.

Após realizar a somatória das entradas com seus pesos aplicados, realizamos a função de ativação:

(1)

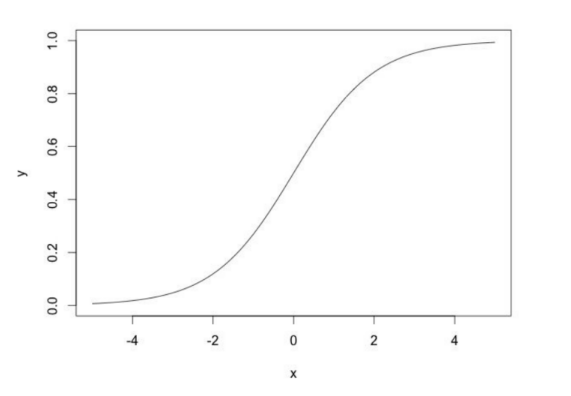


Gráfico 1. Função de Ativação Sigmoide.

Assim, ao final da rede neural, retornando apenas o valor final que define a classificação.

A quantidade de neurônios e em todas as camadas e a quantidade de camada ocultas pode ser alterada, sendo ela definida com base em estudos e testes.

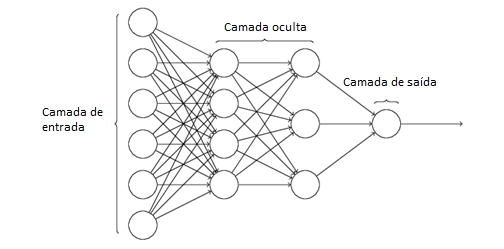


Figura 6. Exemplo de rede neural e suas camadas.

Para a fase de treinamento é utilizado o método *Backpropagation*, onde a mesma realiza ajustes os pesos da última camada assim procedendo para as camadas anteriores, baseado no erro que obteve, ou seja, é necessário informar o valor final desejado para que a mesma verifique se houve ou não erro. Dessa forma o objetivo desse método é de minimizar o erro obtido.

**4.5. *Dataset***

Os *Datasets* serão coletados com pessoas reais e gerados aleatoriamente.

Primeiro foi coletado padrões de pessoas, criando um conjunto de usuários reais e para um aumento na quantidade de usuários, foram criados usuários fictícios. Depois foram criadas tentativas de acessos indevidas aleatórias juntamente com tentativas de clonagem reais. Logo então foi coletada tentativas indevidas de pessoas, de forma aleatória e tentativas de clonagem.

Os dados são passados por normalizações de seus valores. Fórmula de normalização:

(2)

Onde é o valor normalizado, é o valor de entrada que será normalizado, é o menor valor de entrada dos dados do *Dataset* e maior valor.

Para facilitar o processo, os dados gerados pela coleta das assinaturas que são armazenados em um arquivo de texto são processados por um algoritmo em C++ onde ele identifica o maior e menor valor entre todos os valores de seus respectivos tipos de entrada. Então ele percorre os valores e realiza a normalização com a fórmula anterior. Apenas alguns dados são normalizados, pois outros dados podem ser normalizados diretamente durante a coleta devidamente ao fato de podermos pressupor os seus valores máximos e mínimos devido a limitações, assim como a os máximos horizontais e verticais, que é a proporção de seu tamanho original.

Os *dataset* são coletados diretamente do dispositivo móvel, onde ele gera um arquivo de texto para cada usuário e armazena todas suas assinaturas como imagem por motivos de análise.

Para o treinamento da rede neural no ambiente *desktop* os dados são organizados em uma forma viável para a leitura do algoritmo responsável pelo treinamento. Onde os dados apontados pela anotação “*in*” são os dados de entradas e os dados apontados por “*out*” são os dados esperados na saída.

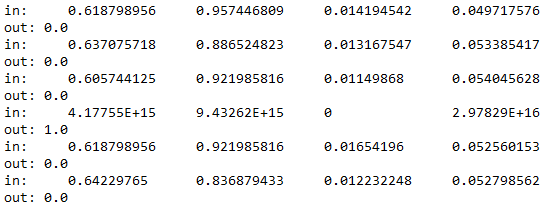


Figura 7. Exemplo dos dados organizados em um arquivo de texto para o treinamento.

**4.6. Funcionamento do sistema**

No formulário principal nós podemos escolher entre três opções:

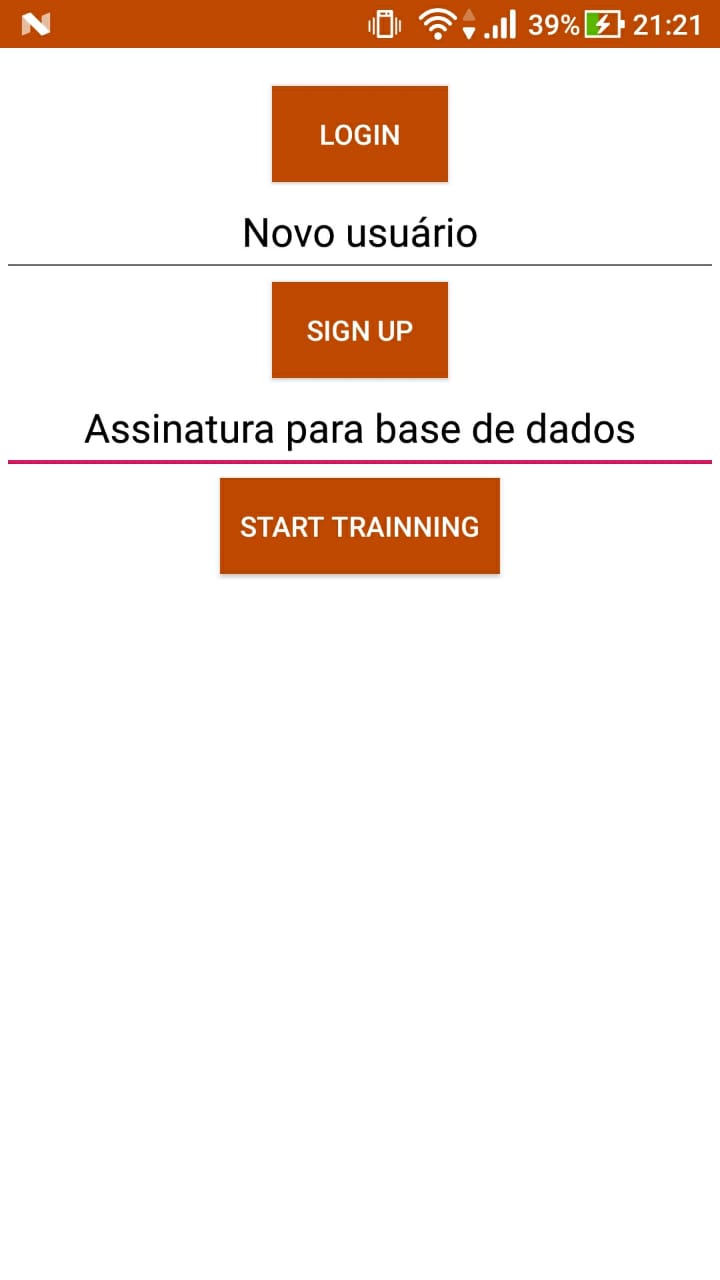


Figura 8. Menu principal da aplicação.

Onde podemos escolher a primeira opção, na qual ela recebe dados uma assinatura como realização do login e conecta a rede neural para o processamento, a fim de verificar se a assinatura é válida ou não. A segunda opção é responsável por abrir o formulário que colhe os dados de um novo cadastro. A terceira opção é responsável pela coleta de dados gerais para utilização no treinamento da rede neural.

Todos os três formulários possuem a mesma aparência exceto por sua *label* que especifica qual formulário foi selecionado.

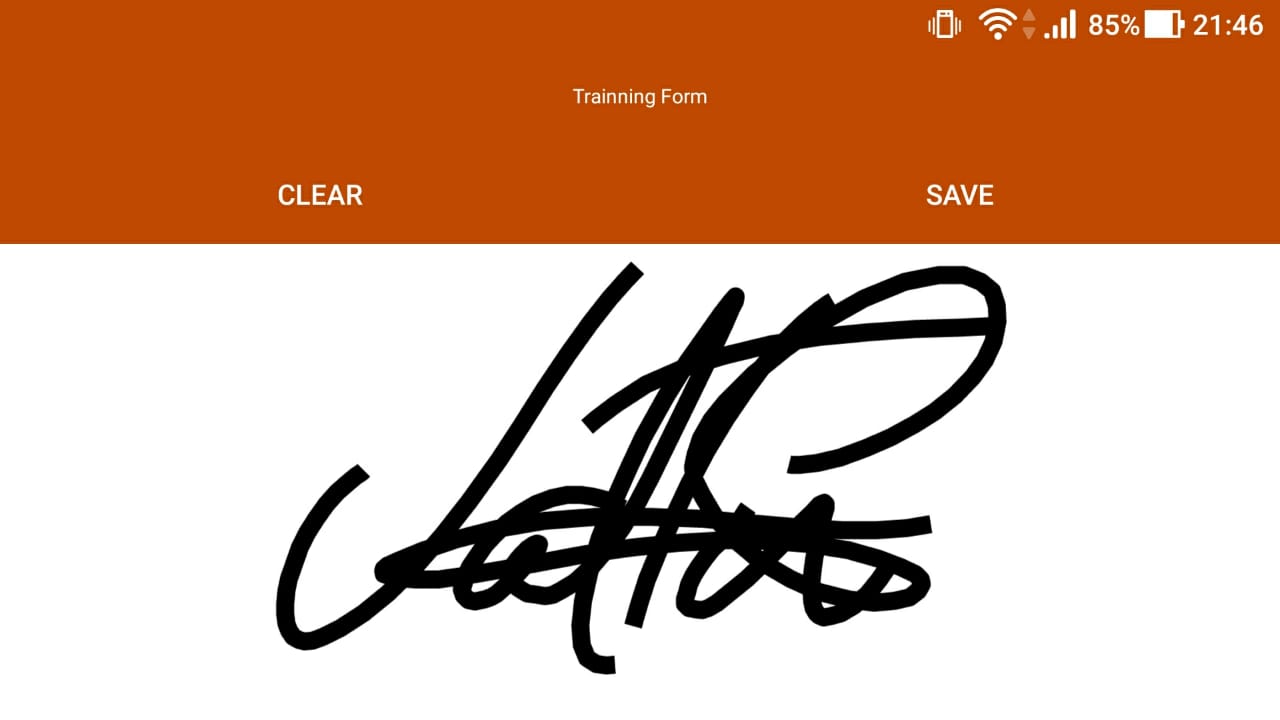


Figura 9. Tela responsável pela coleta da assinatura para treinamento.

**5. RESULTADOS**

Os testes utilizados apresentaram bons resultados, o valor de saída da rede neural necessário para que o usuário seja identificado como cadastrado ou não deve ser maior ou igual que 0,5 para cadastrado e menor que 0,5 para não cadastrado, como primeiro teste foi utilizado um *dataset* reservado para testes no qual o projeto atingiu 80% de acerto, onde não houve acessos indevidos, ou seja, não foi permitido login de nenhum usuário não cadastrado, envolvendo réplicas de assinaturas de usuários cadastrados e 20% acessos devidos negados, usuários cadastrados que tiveram variações relevantes em suas assinaturas e seu login foi negado.

Os próximos gráficos apresentam dados sobre os seguintes resultados: verdadeiro positivo, verdadeiro negativo, falso negativo e falso positivo. Um verdadeiro positivo se refere a um resultado positivo verdadeiro, como um acesso permitido de um usuário cadastrado, verdadeiro negativo é um resultado negativo que era esperado, como uma tentativa de acesso de um usuário não cadastrado que foi negada, falso negativo se refere quando um usuário cadastrado tem seu acesso negado e falso positivo quando um usuário não cadastrado tem seu acesso permitido.

Figura 10. Resultados do *dataset* de testes.

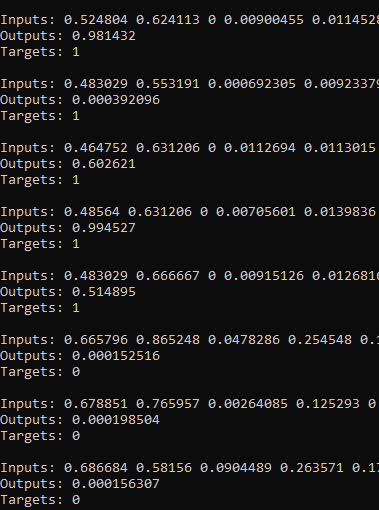


Figura 11. Resultados do *dataset* de testes.

O segundo teste foi realizado em campo, onde foram feitas assinaturas não cadastradas, assinaturas cadastradas e tentativas de réplicas de assinaturas cadastradas. Também obtendo um bom resultado de 90% de acerto, sendo os erros: 3,33% de réplicas de assinaturas cadastradas que foram permitidos login e 6,66% de usuários verdadeiros que tiverem seu login negado.

Figura 12. Resultados do teste de campo.

**6. CONSIDERAÇÔES FINAIS**

Levando em consideração o mesmo ser uma aplicação voltada a segurança, foi optado que ele fique pouco flexível a erros do usuário, por esse motivo alguns usuários têm seus acessos negados mesmo sendo cadastrados, para que o resultado obtenha o mínimo de acessos indevidos possíveis.

O projeto satisfaz a necessidade de um aplicativo de dispositivo móvel que traga tanto facilidade e acessibilidade no uso quanto segurança. Resolvendo os problemas dos usuários que possuem inabilidade de prover os dados necessários para as ferramentas que possuem alta segurança e das empresas que desejam obter soluções viáveis com baixo preço e com mais acessibilidade aos seus funcionários.

O projeto pode ser implementado em aplicações de empresas ou pessoais com necessidade de login, que obtenham as dificuldades descritas anteriormente no uso das ferramentas modernas. Pois em muitos casos essas ferramentas não são viáveis.

**REFERÊNCIAS**

RUSSELL, Stuart; NORVIG, Peter. Inteligência Artificial. 2.ed. Rio de Janeiro: Elsevier, 2004.

HEINEN, Milton Roberto. Autenticação On-line de assinaturas utilizando Redes Neurais. 2002. 92 f. Trabalho de Conclusão de Curso (Bacharel em Informática) - Universidade do Vale do Rio dos Sinos (UNISINOS), Sao Leopoldo, 2002.

GIDEON, S. Jerome et al. Handwritten Signature Forgery Detection using Convolutional Neural Networks. Procedia computer science, v. 143, p. 978-987, 2018.

PANSARE, Ashwini; BHATIA, Shalini. Handwritten signature verification using neural network. International Journal of Applied Information Systems, v. 1, n. 2, p. 44-49, 2012.

DARWAISH, Shah Faisal et al. Biometric identification on android smartphones. Procedia Computer Science, v. 35, p. 832-841, 2014.

CHAUDHRY, Shonal; CHANDRA, Rohitash. Design of a mobile face recognition system for visually impaired persons. arXiv preprint arXiv:1502.00756, 2015.

BINE, Jamilson; NEUMANN, Josiel. Estudo de Segurança em Dispositivos Móveis. Revista Científica Semana Acadêmica Unicentro–PR. Ed, v. 96.

FRIEDLAENDER, Gilda Maria Souza et al. Verificação de assinaturas manuscritas off-line: uma abordagem utilizando modelos estatísticos. 1998.

How to run deep networks on Android device. Disponível em: <https://docs.opencv.org/3.4/d0/d6c/tutorial_dnn_android.html>. Acesso em 16, de maio de 2019

# ROCHA, Marlon de Alencar. OpenCV: Uma breve introdução à visão computacional com python. Disponível em: <https://blog.cedrotech.com/opencv-uma-breve-introducao-visao-computacional-com-python/>. Acesso em: 10, de junho de 2019

Primeiros passos com NDK. Android Developers, 2019. Disponível em: https://developer.android.com/ndk/guides/?hl=pt. Acesso em: 01, de setembro de 2019

A Step by Step Backpropagation Example. Matt Mazur, 2015. Disponível em: <https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/?source=post_page-----bf464f09eb7f---------------------->. Acesso em: 18, de setembro de 2019