



Mineração de Dados Complexos
Curso de Aperfeiçoamento
INF-0619 - Trabalho Final



Sign Language Digits

DeepBlue Team

Daniel Liberman
Rafael Fernando Ribeiro
Thiago Gomes Marçal Pereira

Descrição do Problema

A linguagem de sinais é uma forma de auxiliar deficientes auditivos e da fala a se comunicarem. Para isso pode-se empregar gestos com as mãos, movimento e orientação dos dedos, braços e corpo além de expressões faciais.

Este trabalho consiste em classificar os dígitos de 0 a 9 da linguagem de sinais desenvolvidos pelos gestos da mão de forma correta.

Análise dos Dados

Os dados destes trabalhos estão disponíveis publicamente no site de competições chamado Kagle. Estes dados foram coletados de 218 participantes, cada participante fez o gesto dos números de 0 até 9, sendo assim foram coletadas 10 imagens por participante que é o nosso total de dados disponível. As imagens foram coletadas por uma câmera Canon em local fechado, com um fundo branco e ambiente controlado, elas possuem resolução de 100x100 pixels e foram armazenadas em formato RGB, ou seja coloridas.

A figura 1 apresenta um conjunto de imagens expressando os números de 0 até 9.

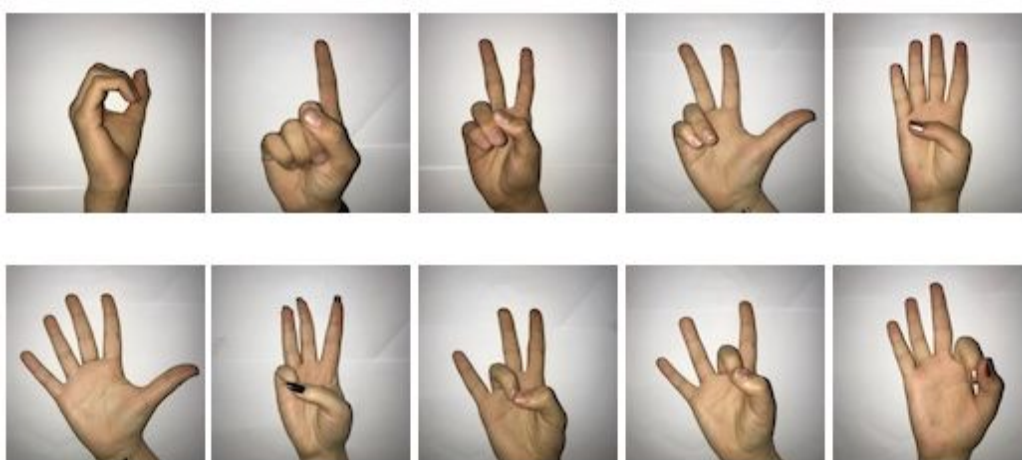
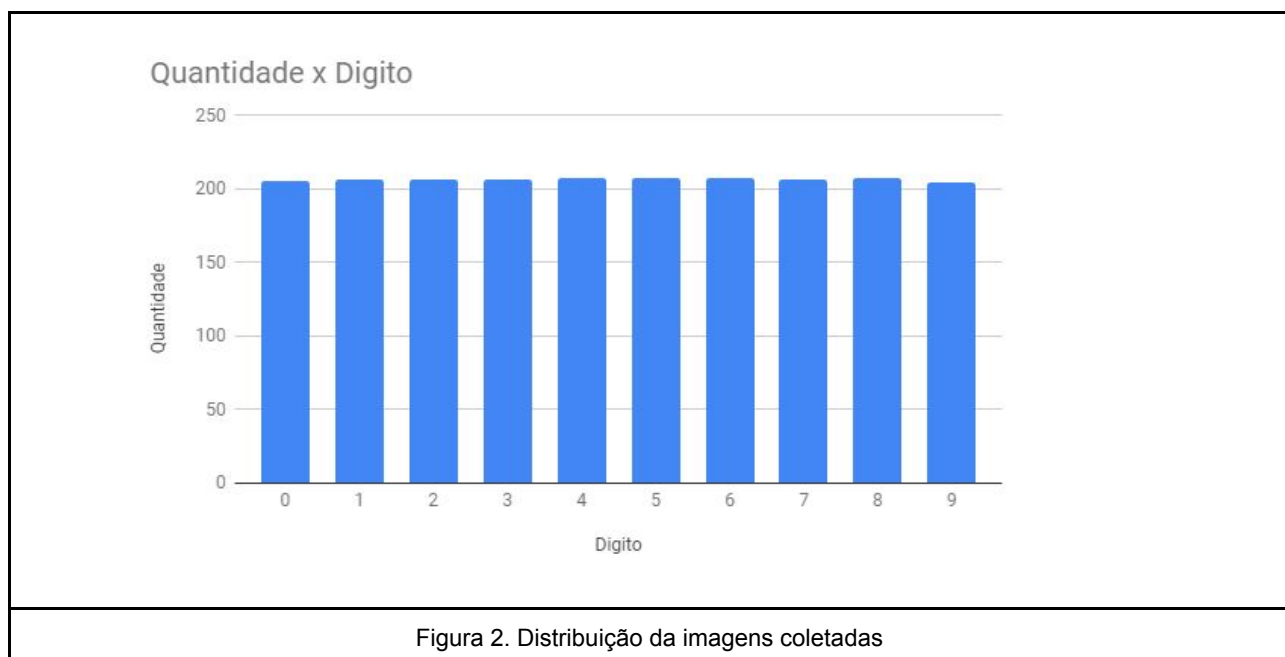


Figura 1, apresentação dos números de 0-4 na primeira linha e de 5-9 na segunda.

Os dados foram coletados de forma regular e estão distribuídos de maneira que a quantidade de dados de cada classe é muito similar umas às outras, conforme é apresentado na figura 2. mostrando que nossa base de dados de imagens está balanceado, sendo que o número máximo de imagens por classe é 208 e o número mínimo é 204, nesse caso não é necessário utilizar nenhuma técnica para fazer o balanceamento dos mesmos. No total temos 2062 imagens de 0 até 9.



Pré-Processamento dos dados

As imagens foram capturadas em ambiente controlado e todas as imagens tem um plano de fundo branco, o que facilita no nosso problema. Uma análise não mostrou nenhuma inconsistência na base, como imagens como valores trocados e imagens sem algum nexo referente ao problema. As imagens das mãos se encontram centralizadas o que também é bom para nosso desenvolvimento. As imagens foram carregadas em formato RGB e preservamos todos os canais de cores.

Para os modelos utilizando redes neurais usados os dados da forma como foram armazenados e também aplicamos o conceito de Data Augmentation (aumento de dados) em outros testes. Utilizamos as seguintes técnicas para aumentar os dados:

Transformação dos dados

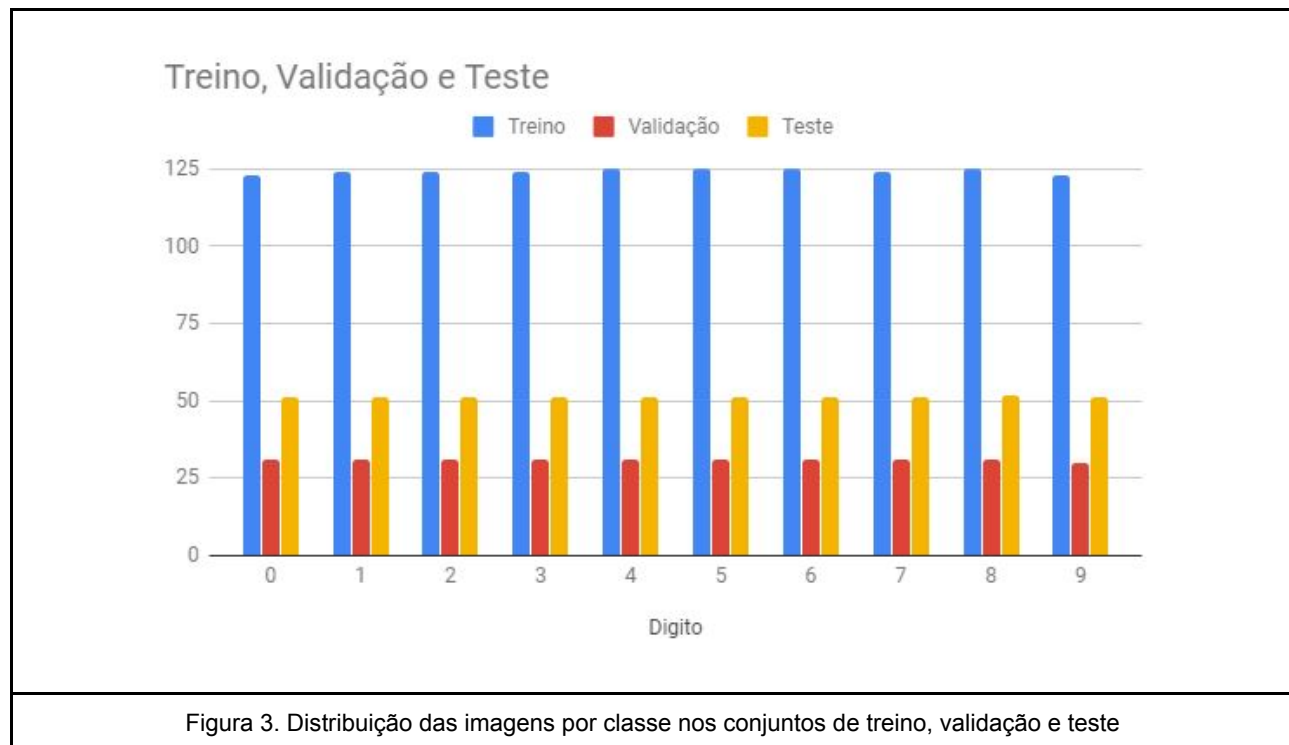
Por se tratar de imagens, elas são representadas por pixels que variam de 0 até 255 e isso serve para todas as imagens, sendo assim não é necessário nenhuma forma de normalização dos dados, podemos utilizá-los da forma como estão, no máximo em alguns modelos fizemos a divisão dos valores por 255 para que os dados fiquem na faixa de 0 a 1.

A seleção de características foram feitas todas de forma automática, visto que isto é uma tendência à se utilizar nos problemas de classificação de imagens, pois estes métodos apresentam resultados superiores a extração de características de forma manual.

Utilizamos primeiramente alguns classificadores clássicos como Regressão Logística, SVM e etc com características retiradas da última camada de uma rede neural pré treinada no caso a ResNet50, com isso obtivemos um vetor de características de tamanho 2048.

Não foi utilizado nenhuma forma para reduzir os dados em nosso trabalho visto que os modelos apresentaram bom desempenho da forma com que os dados foram apresentados.

Por último explicaremos como foi o particionamento de nossa base de dados de forma a criar o conjunto de imagens para treinamento, validação e teste. Neste quesito fizemos a distribuição da seguinte forma, 60% das imagens para treino, 15% para validação e 25% para teste, resultado em 1242 imagens para treino com uma média de 124 imagens por dígito, 309 para validação com aproximadamente 31 de cada dígito e 511 para teste resultado em 51 por dígito.



Modelagem

Nosso problema envolve a classificação dos dígitos da linguagem de sinais, neste caso temos todas as imagens rotuladas com o respectivo número de correspondência, sinalizando que podemos trabalhar de forma supervisionada para realizar a classificação.

No que diz respeito aos modelos abordados, fizemos uma seleção de modelos mais clássicos como Regressão Logística, kNN, SVM e Florestas Randômicas e depois passamos a abordar modelos mais complexos que se baseiam em Redes Neurais Convolucionais. Iremos descrever a abordagem realizada para teste de cada modelo e no final teremos o embasamento necessário para a escolha do melhor modelo.

Como foi dito anteriormente para todas as simulações envolvendo as técnicas clássicas as características foram obtidas da extração das mesmas de uma rede neural convolucional previamente treinada com as imagens da base conhecida ImageNet. Isso resultou em um vetor de características com tamanho de 2048. A figura 4 detalha o processo de extração de características.

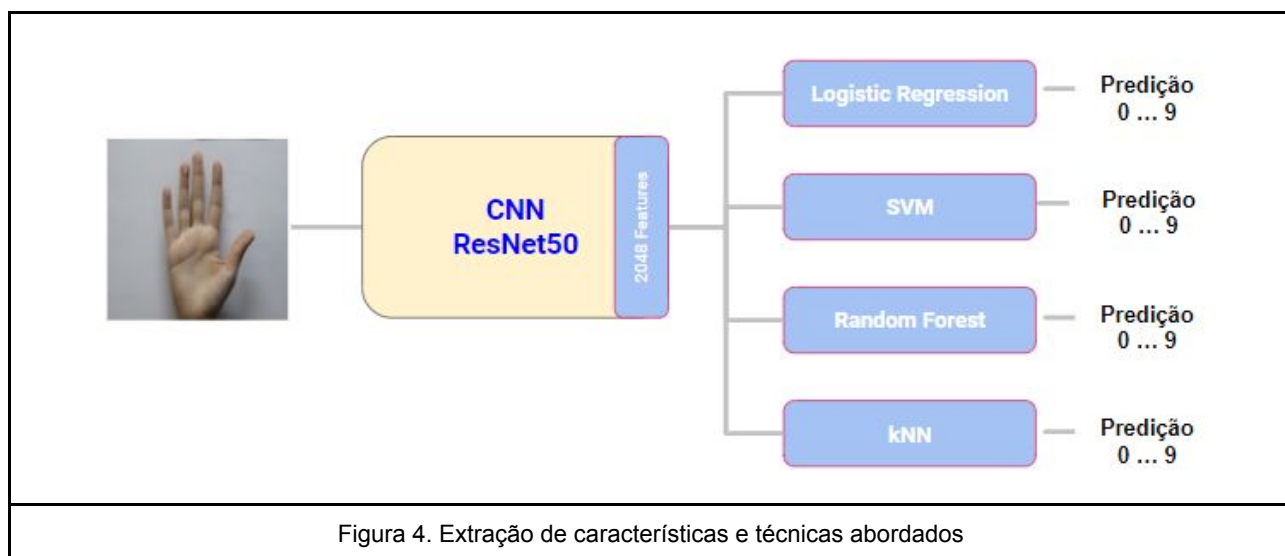


Figura 4. Extração de características e técnicas abordados

Para todos os modelos testados foi utilizada a técnica de grid search para encontrar os melhores hiperparâmetros que melhor se adaptassem ao problema. A lista dos parâmetros testados será apresentado em cada modelo abordado. Utilizamos a estratégia de validação cruzada com 2 folders.

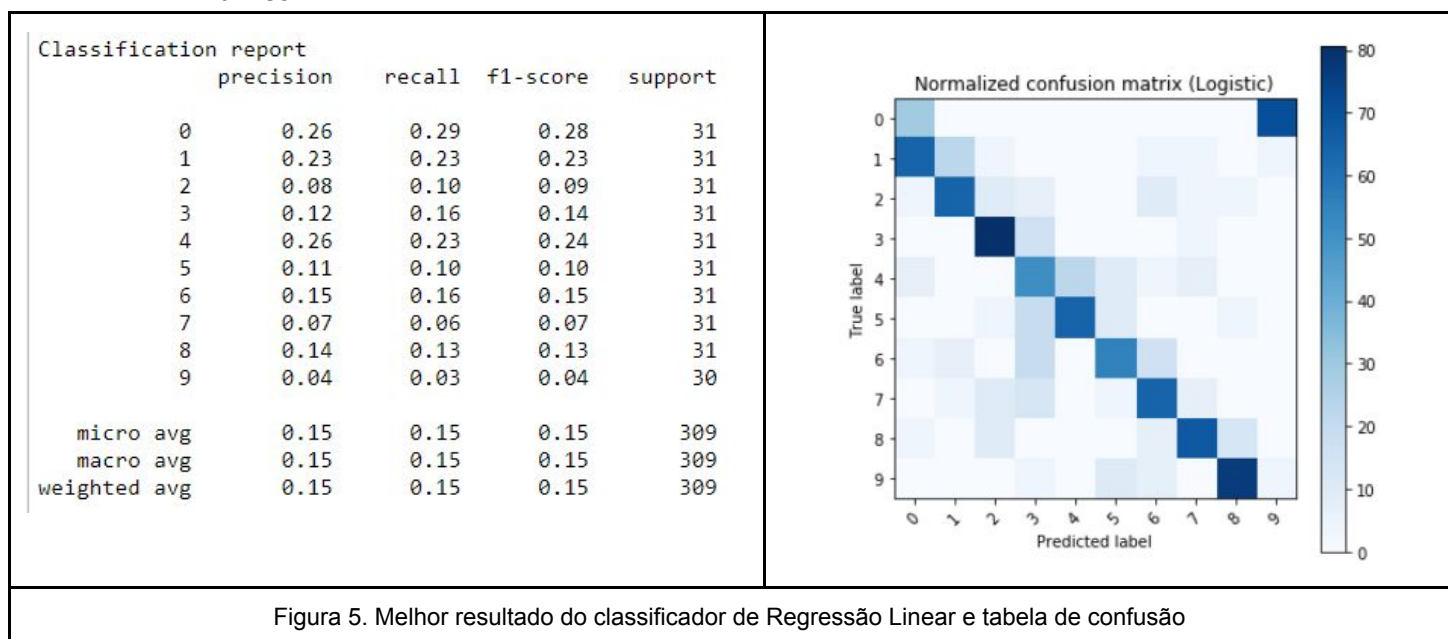
Regressão Logística:

Para a regressão logística foi testado com os seguintes parâmetros para a variável:

$C = 0.01, 0.1, 1.0, 1, 10, 50, 100, 500, 1000$

solvers = 'lbfgs', 'liblinear'.

Obtivemos 29% de acurácia no melhor resultado utilizando os parâmetros $C=50$ e solver='liblinear'



K-Nearest Neighbors:

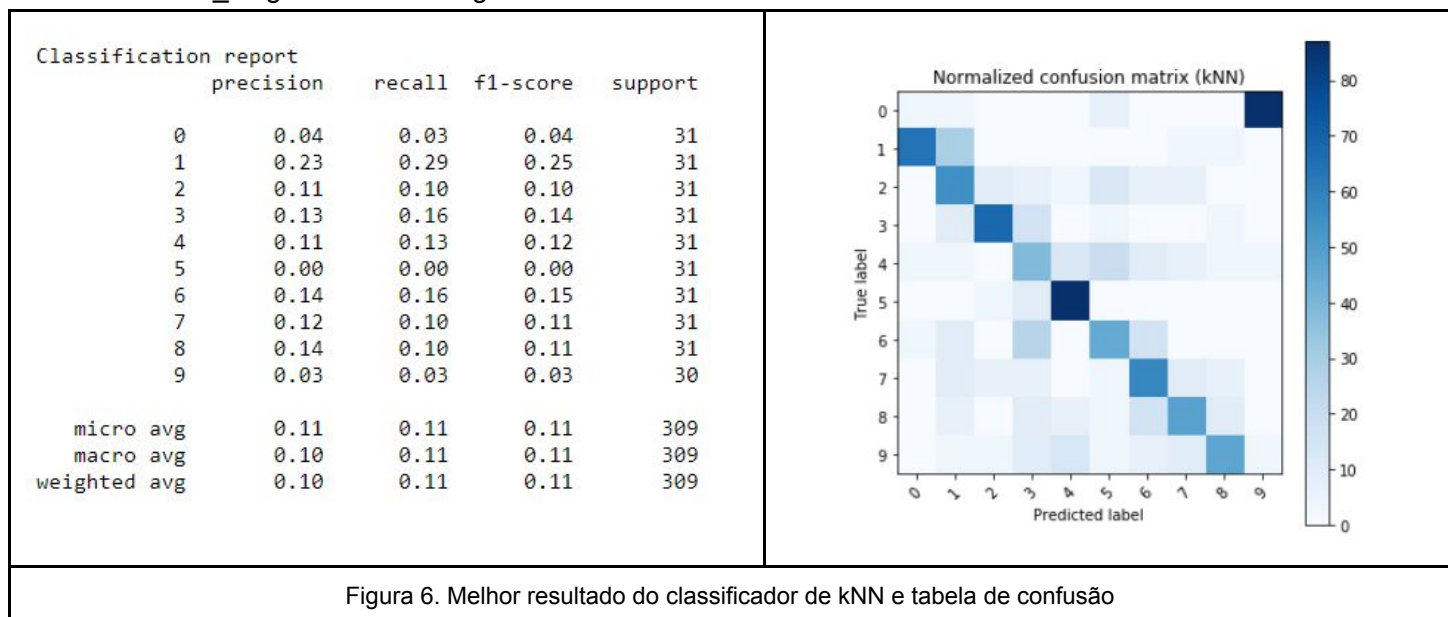
Para este modelo testamos com os seguintes parâmetros para a variável:

$n_neighbors = [3, 5, 7, 10]$

weights = ['uniform', 'distance']

algorithm = ['auto', 'kd_tree', 'brute']

Obtivemos 25% de acurácia no melhor resultado utilizando os parâmetros `algorithm='auto'`, `n_neighbors=10` e `weights = 'distance'`



Support Vector Machine:

Para este modelo testamos com os seguintes parâmetros para a variável:

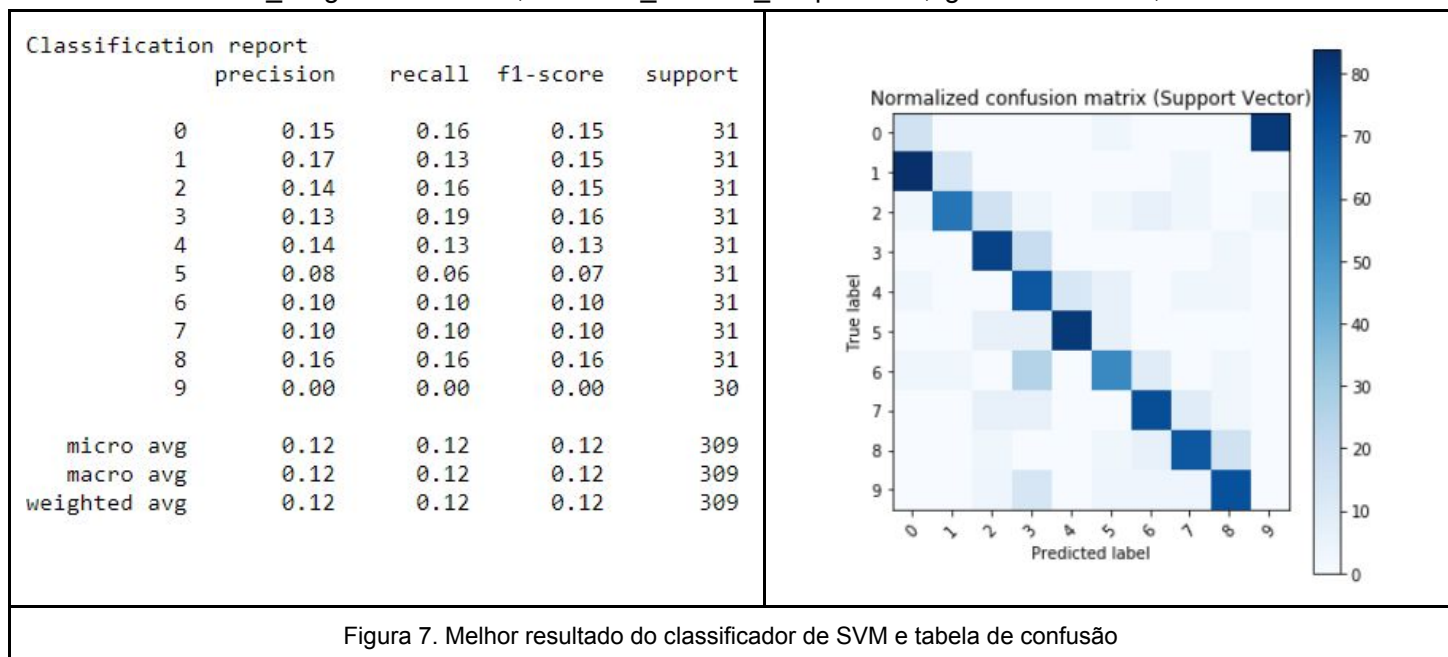
`'kernel': ['rbf']`

`'gamma': [0.01, 0.001, 0.0001]`

`'decision_function_shape':['ovo']`

`'C': [0.01, 0.1, 1.0, 1, 10, 50, 100, 500, 1000]`

Obtivemos 29% de acurácia no melhor resultado utilizando os parâmetros `'C'=500`, `'class_weight'= 'balanced'`, `'decision_function_shape'='ovo'`, `'gamma'= 0.0001`, `'kernel'= 'rbf'`



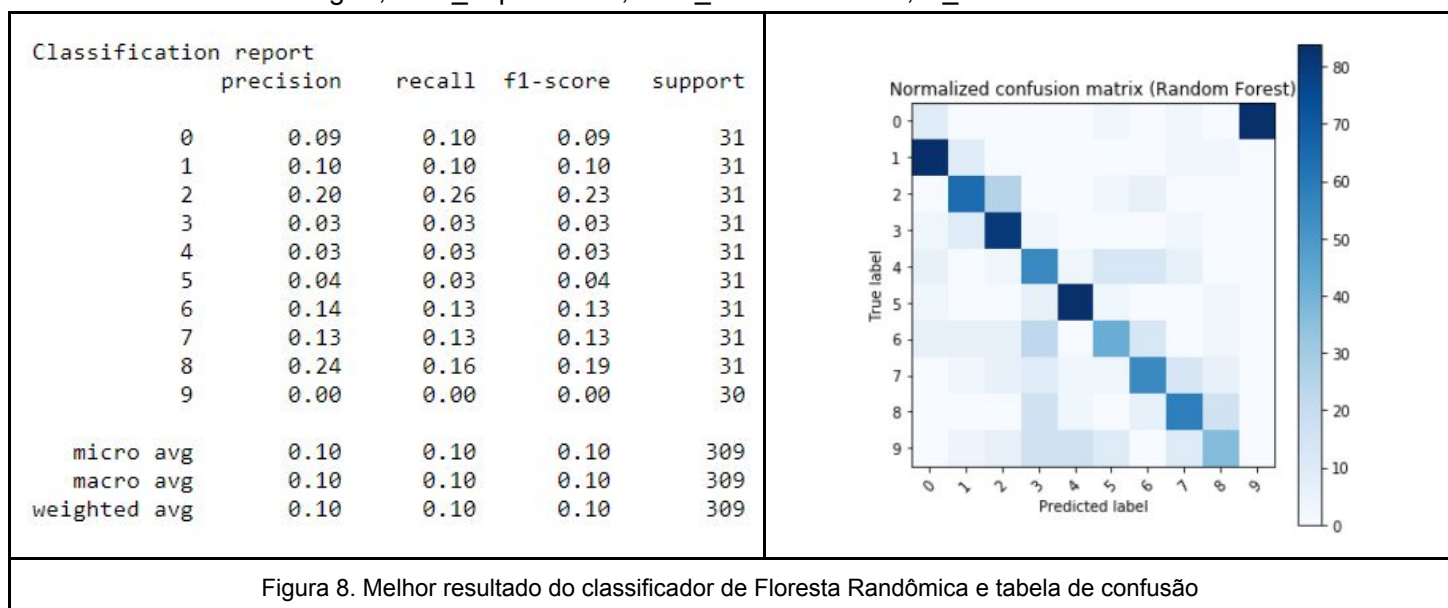
Random Forest:

Para este modelo testamos com os seguintes parâmetros para a variável:

`'n_estimators': [10, 20, 50, 100,200]`,

```
'max_features': ['auto', 'log2'],
'max_depth': [None],
'bootstrap': [True, False],
'criterion': ["gini", "entropy"]
```

Obtivemos 24% de acurácia no melhor resultado utilizando os parâmetros: 'bootstrap'=True, 'criterion'='gini', 'max_depth'=None, 'max_features'= 'auto', 'n_estimators'= 50



Como podemos observar nos modelos apresentados, a acurácia não ficou acima de 30% e nenhuma das técnicas utilizadas. Baseado neste contexto iniciamos nossa abordagem dos modelos baseados em redes neurais convolucionais. Utilizamos as redes ResNet50, MobileNetV2, SqueezeNet, para estas redes utilizamos os modelos previamente treinados na base de dados da ImageNet e fizemos um processo de transferência de aprendizado para nosso problema, onde aplicamos uma taxa de aprendizado pequena e refinamos todas as camadas da rede. Também utilizamos uma Rede Neural Siamesa para nossos testes.

Todos os testes foram realizados durante 30 épocas, porém utilizamos técnica de “early stop” que termina o treino caso o modelo não apresente uma melhora depois de 3 épocas. O tamanho do “batch” é dependente da máquina utilizada, no nosso caso a máquina tinha uma GPU de 4GB e utilizamos o tamanho de 12 imagens por batch.

SqueezeNet:

Para este modelo testamos com os seguintes parâmetros dos otimizadores:

Adam com learning rate = 0.00001

Adadelta com learning rate = 1.0

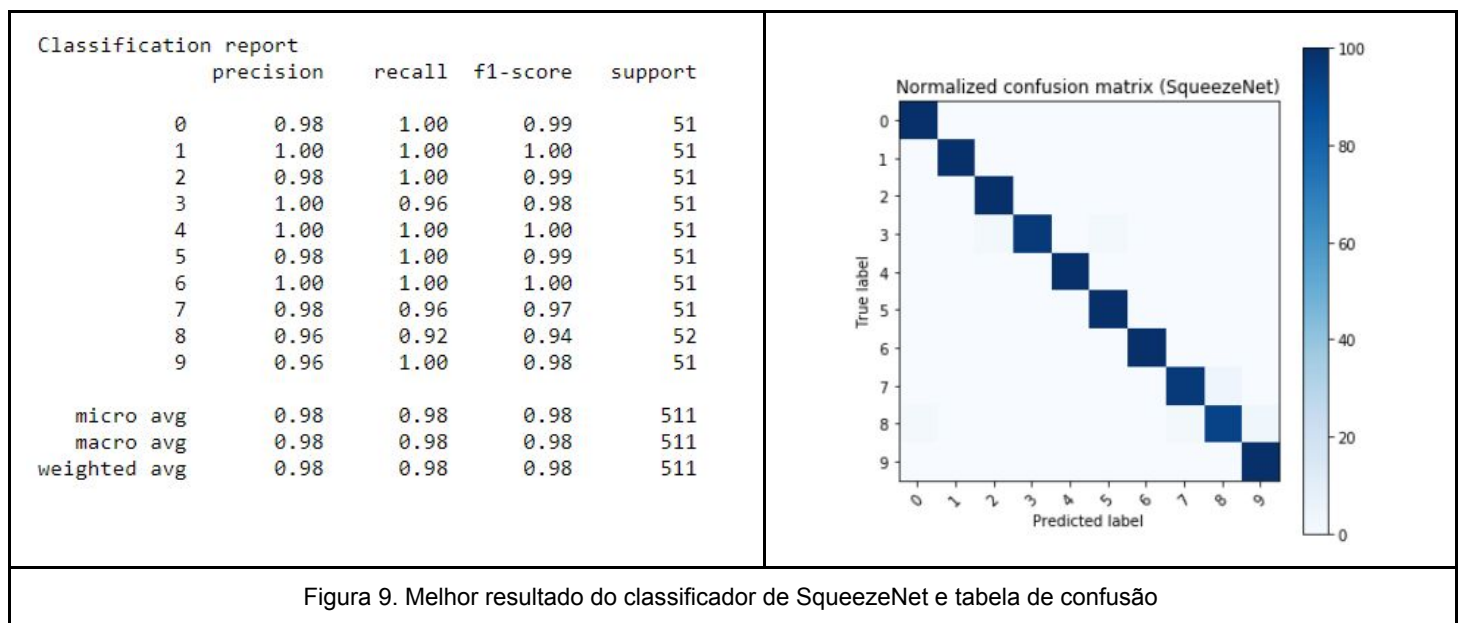
SGD com learning rate = 0.001 e momento = 0.9

Tamanho da imagem de input: 100x100

Os resultados no conjunto de teste foram:

Otimizador	Sem Data Augmentation	Com Data Augmentation
Adam	95,97 %	95,83%
Adadelta	98,38 %	98,41%

SGD	98,58 %	96,03%
-----	---------	--------



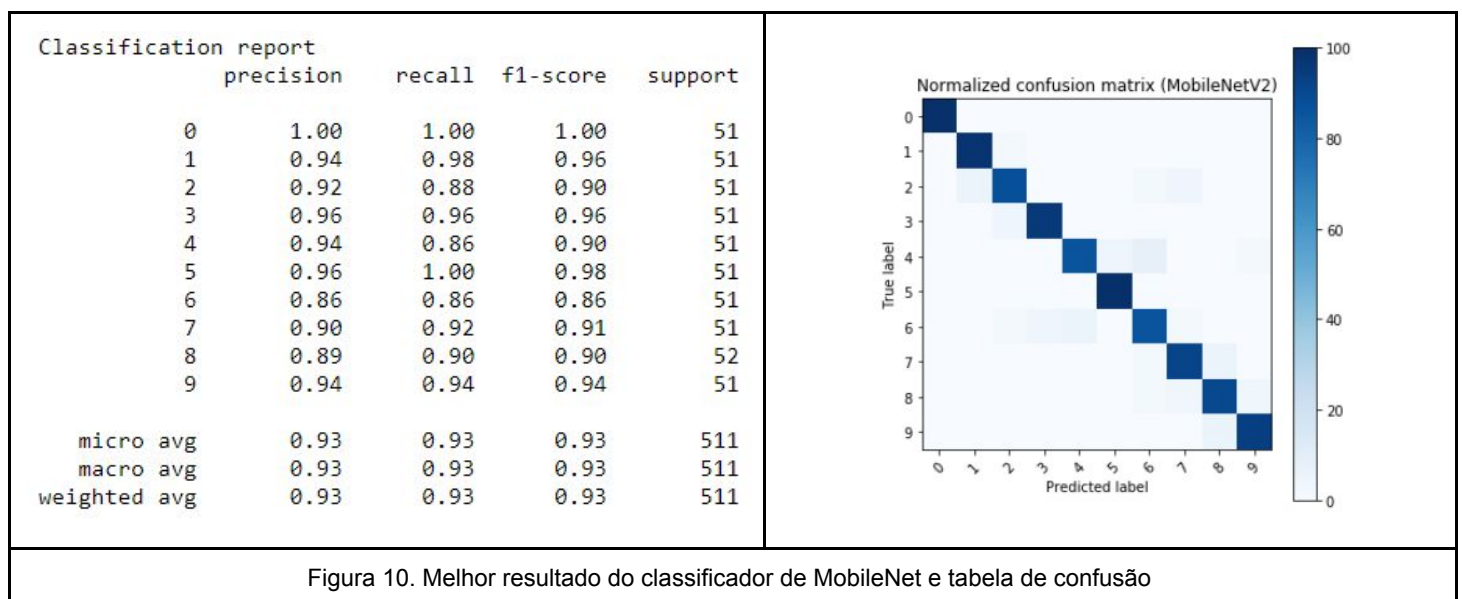
MobileNet:

Para este modelo testamos com os seguintes parâmetros do otimizador:

Adam com learning rate = 0.001

Os resultados no conjunto de teste foram:

Otimizador	Sem Data Augmentation	Com Data Augmentation
Adam	92,2 %	92,12 %



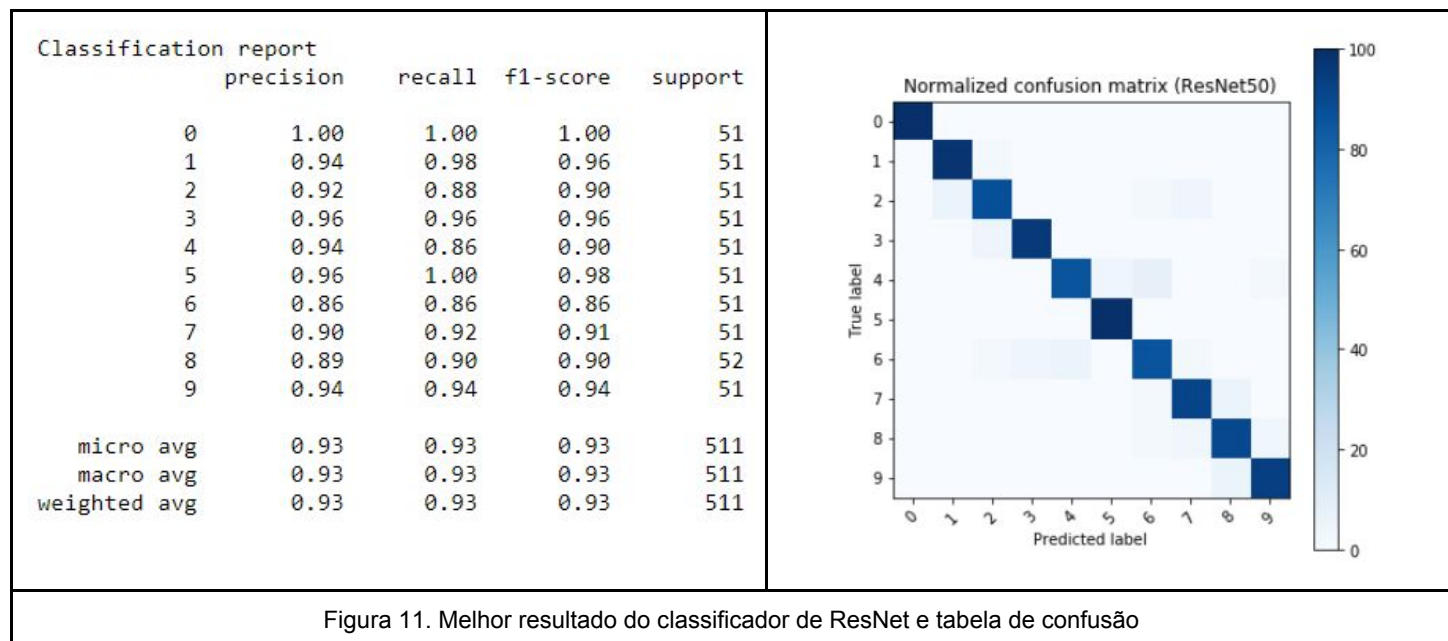
ResNet50:

Para este modelo testamos com os seguintes parâmetros do otimizador:

Adam com learning rate = 0.001

Os resultados no conjunto de teste foram:

Otimizador	Sem Data Augmentation	Com Data Augmentation
Adam	90,55 %	95,67 %



Rede Siamesa:

Para este modelo testamos com os seguintes parâmetros do otimizador:

Adam com learning rate = 0.001

Os resultados no conjunto de teste foram:

Otimizador	Sem Data Augmentation	Com Data Augmentation
Adam	97,39 %	97,11 %

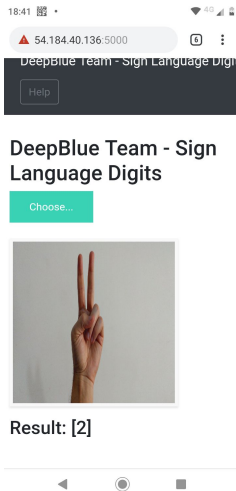
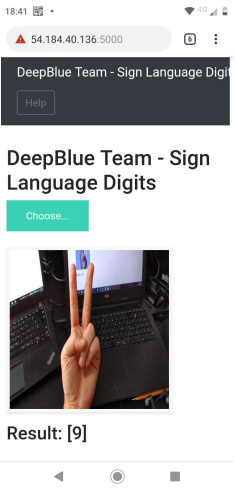
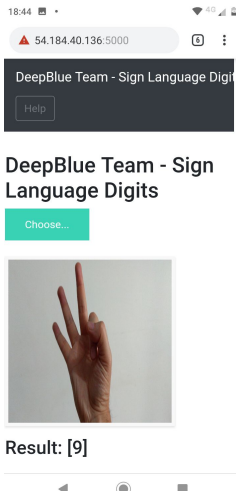
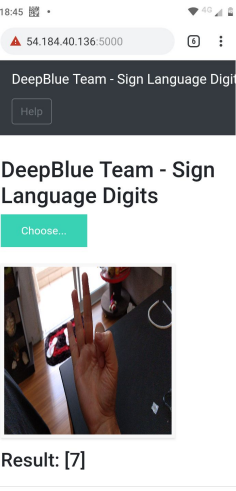
Demo

Após o treino dos nossos modelos, escolhemos o modelo que apresentou os melhores resultados e criamos um sistema que funciona tanto na web quanto no celular para carregar uma imagem e classificá-la conforme o dígito apresentado.

A demo pode ser acessada pelo endereço: <http://http://54.184.40.136:5000/>

Ela apresentou uma generalização muito boa e não comete nenhum erro nas imagens da base de dados fornecidas.

Resolvemos apresentar novas imagens coletadas a partir do celular para o modelo, neste caso o modelo apresentou algumas falhas conforme podemos ver na tabela abaixo. Este problema acontece principalmente por causa do tamanho da imagem coletada, que no caso do celular pode chegar 3264x1546 pixels, necessitando um pré processamento, outro fator em questão é o fundo da imagem, pois como o banco de dados original foi criado em um ambiente controlado, quando utilizamos imagens sem controle do ambiente e com ruído ao fundo o classificador não está preparado para tratar estes casos.

Casos de sucesso	Casos de erros
	
	

Conclusão

Chegamos a algumas conclusões sobre este trabalho, primeiro que os modelos clássicos obtiveram um desempenho muito ruim comparado com os modelos baseados em redes neurais, comprovando uma tendência da área de visão computacional. Segundo, apesar de nossa base de dados não ser muito grande (125 imagens por classe para treino) comparando com outras bases existentes e utilizadas para treino de redes neurais, conseguimos um ótimo resultado utilizando a técnica de transferência de aprendizado e aumento de imagens utilizando data augmentation (translações, rotações e escala) em que a acurácia ficou superior a 98% com um f1-score de 98%, o que resultou em um modelo robusto e com uma boa generalização para este tipo de problema. O nosso melhor modelo foi a SqueezeNet que alcançou a melhor acurácia e f1-score, esta rede é ainda mais interessante pois ela possui um total de 1.258.058 parâmetros que consomem apenas 10Mb de dados em disco e com isso ela também tem um desempenho mais rápido em sua execução.

A aplicação demonstrativa gerada a partir deste trabalho mostrou que nosso modelo assim como todos os outros ainda não é perfeito e sofre com alguns problemas devido ao ambiente controlado em que as imagens foram coletadas, acredito que com mais tempo e algumas técnicas de aumento de dados podemos alcançar um melhor resultado nestes ambientes mais realistas.