

# **Processamento de Imagens - IF69P/C81**

Trabalho Final

Transfer Learning aplicado a arquiteturas CNN

**Vinicius Augusto de Souza - 1997530**

Engenharia de Computação

# Introdução

# Introdução

O projeto tem como objetivo aplicar técnicas de *transfer learning* a diferentes arquiteturas CNNs por meio do treinamento realizado no dataset ImageNet.

- A linguagem de programação utilizada foi Python
- Foi utilizado o *Jupyter Notebook* para realizar todo o projeto
- A arquitetura CNN escolhida foi a ResNet50V2
- Para os experimentos, foi utilizado 80% do dataset para treino e 20% para teste

# Introdução

As bibliotecas utilizadas no projeto foram:

- random
- os
- tqdm
- shutil
- pathlib
- tensorflow
- numpy
- pandas
- sklearn
- matplotlib
- seaborn
- os

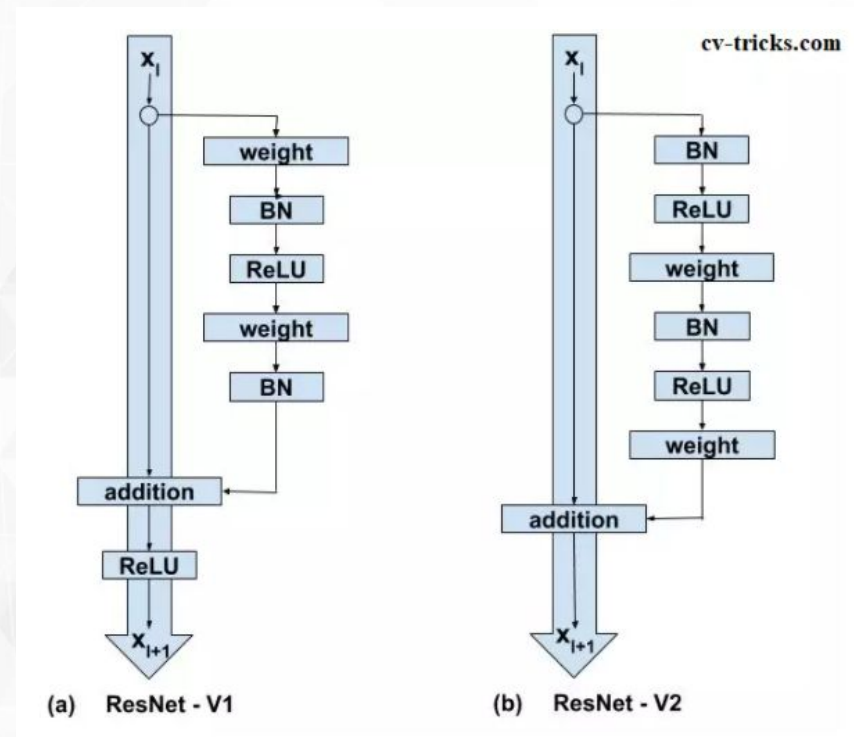
# Conceitos e Trabalhos relacionados



## Arquitetura ResNet50V2

ResNet50V2 é uma versão modificada do ResNet50 e seu desempenho é melhor do que ResNet50 e ResNet101 no conjunto de dados ImageNet. No ResNet50V2, o modo de propagação das conexões entre os blocos foi modificado. ResNet50V2 também obteve bons resultados no conjunto de dados ImageNet.

# Arquitetura ResNet50V2



FONTE: <https://cv-tricks.com/keras/understand-implement-resnets/>

# Arquitetura ResNet50V2

- Depois que o ResNet V1 executa uma operação de adição entre  $x$  e  $F(x)$ , ele adiciona a segunda não linearidade. O ResNet V2 elimina a última não linearidade, portanto, o caminho da entrada à saída é eliminado na forma de conexões de identificação.
- Antes de multiplicar pela matriz de peso (operação de convolução), o ResNet V2 aplica a normalização de lote e a ativação de ReLU à entrada. O ResNet V1 realiza a convolução e, em seguida, normaliza em lote e ativa o ReLU.



# Arquitetura ResNet50V2

ResNet - V1	ResNet - V2
$y = x_l + F(x_l, \{W_{ij}\})$ $x_{l+1} = H(x) = \text{ReLU}(y)$	$y = h(x_l) + F(x_l, \{W_{ij}\})$ $x_{l+1} = H(x) = f(y)$
$y = \text{Addition Output}$ $x_{l+1} = \text{Input to Next Block}$	$y = \text{Addition output}$ $h(x_l) = \text{Generalized form of input.}$ For ResNet V1, $h(x_l) = x_l$ .  $f = \text{Function applied to 'y'}$ . For ResNet V1, $f = \text{ReLU}$ .  For ResNet V2, $f$ is an identity mapping.

**FONTE:** <https://cv-tricks.com/keras/understand-implement-resnets/>

## Características principais da arquitetura ResNet50V2

- O ResNet usa a normalização em lote em seu núcleo. A normalização em lote pode ajustar a camada de entrada para melhorar o desempenho da rede. O problema de mudar as variáveis é reduzido.
- Utiliza a Conexões de Identidade para ajudar a proteger a rede de problemas de gradiente de desaparecimento.
- A rede residual profunda usa o projeto do módulo de gargalo residual para melhorar o desempenho da rede.

# Descrição da Base de Imagens

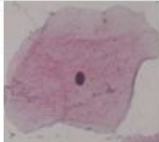

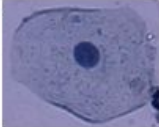



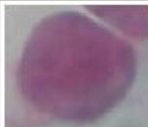
# Dataset PapSmear

O teste de Papanicolaou, também conhecido como teste de Papanicolaou, é um procedimento para testar mulheres quanto ao câncer cervical. O câncer cervical é um tipo de câncer que ocorre nas células do colo do útero, que são a parte inferior do útero que conecta a vagina.

Este dataset consiste em 917 amostras distribuídas desigualmente em 7 classes diferentes, das quais são classificadas como células normais as *Superficial squamous*, *Intermediate squamous* e *Columnar* e as células anormais, que são as classes *Mild dysplasia*, *Moderate dysplasia*, *Severe dysplasia* e *Carcinoma in situ*.

Norup, Jonas. "Classification of Pap-smear data by transduction neuro-fuzzy methods." (2005).

# Dataset PapSmear

Normal cells		Abnormal cells	
<b>Superficial squamous 1</b> <ul style="list-style-type: none"> <li>● Shape: Flat/oval</li> <li>● Nucleus very small</li> <li>● N/C very small</li> </ul>			<b>4 Mild dysplasia</b> <ul style="list-style-type: none"> <li>● Nucleus light/large</li> <li>● N/C medium</li> </ul>
<b>Intermediate squamous 2</b> <ul style="list-style-type: none"> <li>● Shape: Round</li> <li>● Nucleus large</li> <li>● N/C small</li> </ul>			<b>5 Moderate dysplasia</b> <ul style="list-style-type: none"> <li>● Nucleus large/dark</li> <li>● Cytoplasm dark</li> <li>● N/C large</li> </ul>
<b>Columnar 3</b> <ul style="list-style-type: none"> <li>● Shape: Column-like</li> <li>● Nucleus large</li> <li>● N/C medium</li> </ul>			<b>6 Severe dysplasia</b> <ul style="list-style-type: none"> <li>● Nucleus large/dark/deform</li> <li>● Cytoplasm dark</li> <li>● N/C very large</li> </ul>
			<b>7 Carcinoma in situ</b> <ul style="list-style-type: none"> <li>● Nucleus large/dark/deform</li> <li>● N/C very large</li> </ul>

Norup, Jonas. "Classification of Pap-smear data by transduction neuro-fuzzy methods." (2005).



# Separação do Dataset

```
data_dir = "Database"
classes = ["carcinoma_in_situ", "light_dysplastic", "moderate_dysplastic", "normal_columnar",
"normal_intermediate", "normal_superficial", "severe_dysplastic"]
output_dir = "data_tt_g"
ratio = [0.8, 0.2]

def split(data_dir, output_dir, ratio):
    for cell in classes:
        cell_path = os.path.join(data_dir, cell)
        files = os.listdir(cell_path)
        files = [os.path.join(cell_path, f) for f in files if f.endswith('.BMP')]

        random.seed(230)
        files.sort()
        random.shuffle(files)

        split_train = int(ratio[0] * len(files))
        split_test = split_train

        files_train = files[:split_train]
        files_test = files[split_train:]
        files_type = [(files_train, "train"), (files_test, "test")]

    for (files, folder_type) in files_type:
        full_path = os.path.join(output_dir, folder_type)
        full_path = os.path.join(full_path, cell)
        pathlib.Path(full_path).mkdir(parents=True, exist_ok=True)
        for f in files:
            shutil.copy2(f, full_path)

split(data_dir, output_dir, ratio)
```

# Estrutura e Conceitos

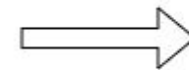
## Estrutura

- Carrega-se o modelo da ResNet50V2 com os pesos aprendidos no treino da ImageNet sem a camada densa
- Adiciona-se um nó utilizando o método de *GlobalAveragePooling*
- Adicionam-se nós de camada densa com o método de ativação ReLu contendo:
  - 128 neurônios
  - 64 neurônios
  - 32 neurônios
- Adiciona-se uma camada de Dropout, desligando uma porcentagem de neurônios
- Por fim, uma última camada densa, com 7 neurônios (número de classes) com a função de ativação *softmax*
- Foi utilizado o método *EarlyStopping*, no qual para o treinamento quando uma métrica monitorada parar de melhorar

## GlobalAveragePooling

- Ao aplicar o método, o tamanho do pool ainda é definido para o tamanho da entrada da camada, e a média é definida
- Eles são frequentemente usados para substituir as camadas totalmente conectadas ou densamente conectadas em um classificador.
- Ao alimentar os valores gerados pelo agrupamento médio global em uma função de ativação do Softmax, você obtém mais uma vez a distribuição de probabilidade multiclasse que deseja.

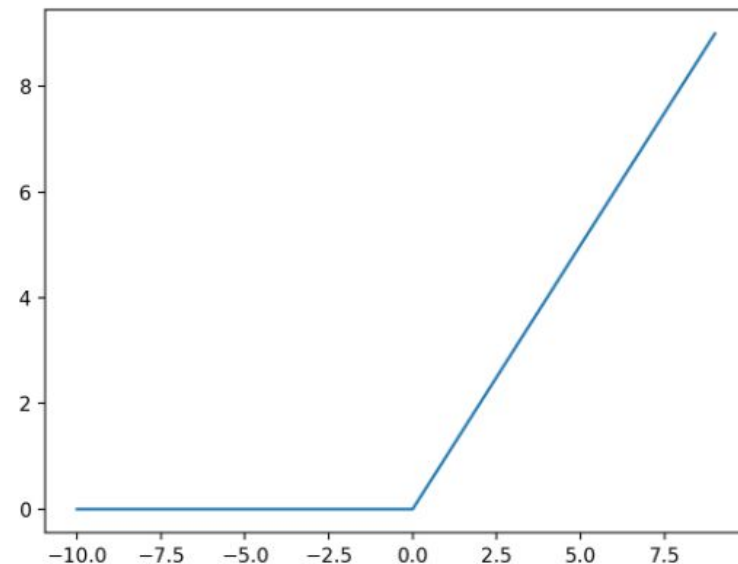
4	3	1	5
1	3	4	8
4	5	4	3
6	5	9	4



4.3

## ReLu

- Em uma rede neural, a função de ativação é responsável por transformar a entrada ponderada somada do nó na ativação do nó ou saída para essa entrada.
- A função de ativação linear retificada ou ReLU para breve é uma função linear por partes que produzirá a entrada diretamente se for positiva, caso contrário, ela produzirá zero.





# Softmax

- Softmax é uma função matemática que converte um vetor de números em um vetor de probabilidades, onde as probabilidades de cada valor são proporcionais à escala relativa de cada valor no vetor.
- É utilizada como a função de ativação na camada de saída de modelos de rede neural que prevêem uma distribuição de probabilidade multinomial.

Formula

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}}$$

$\sigma$  = softmax

$\vec{z}$  = input vector

$e^{z_i}$  = standard exponential function for input vector

$K$  = number of classes in the multi-class classifier

$e^{z_j}$  = standard exponential function for output vector

$e^{z_j}$  = standard exponential function for output vector

# Cenários

## Cenários

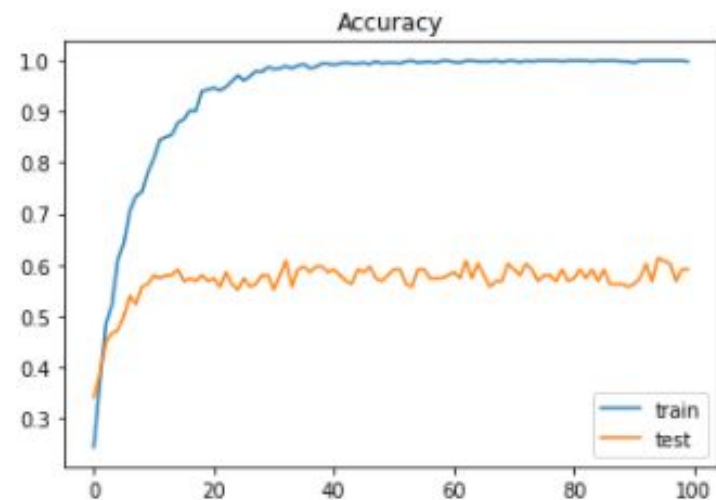
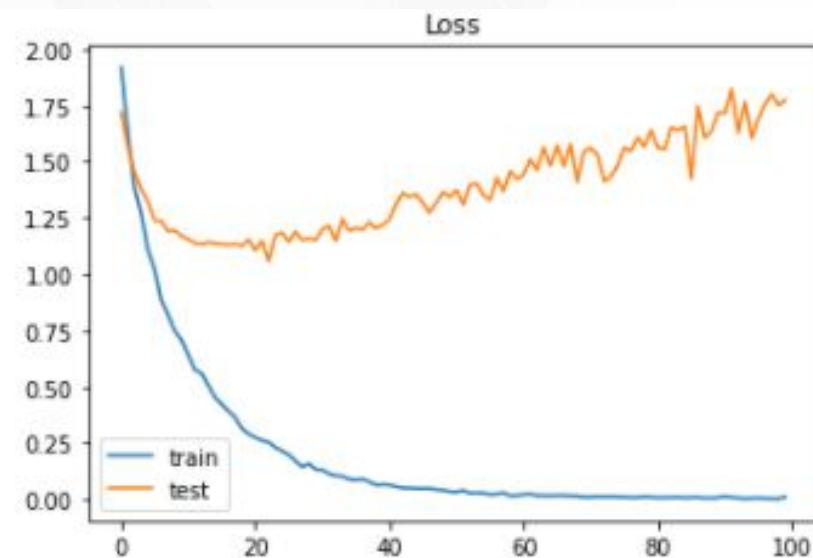
Foram realizados 8 testes, nos quais foram alterados os seguintes parâmetros

- *batch* - número de amostras que será carregado a cada execução
- *dropout* - desliga uma porcentagem de neurônios
- *learning\_rate* - é um parâmetro de ajuste em um algoritmo de otimização que determina o tamanho da etapa em cada iteração enquanto se move em direção a um mínimo de uma função de perda.

Por padrão, foram utilizadas 100 *epochs* e 128x128 de tamanho de imagem

**batch: 16, dropout: 0.1, learning\_rate: 0.0001**

Train: 1.000, Test: 0.591





# batch: 16, dropout: 0.1, learning\_rate: 0.0001

Train: 1.000, Test: 0.591

## -----CLASSIFICATION-----

	precision	recall	f1-score	support
--	-----------	--------	----------	---------

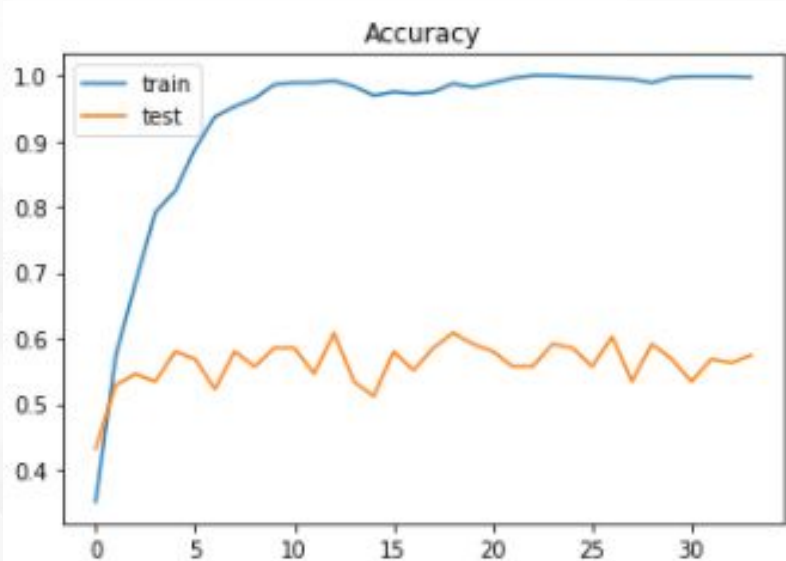
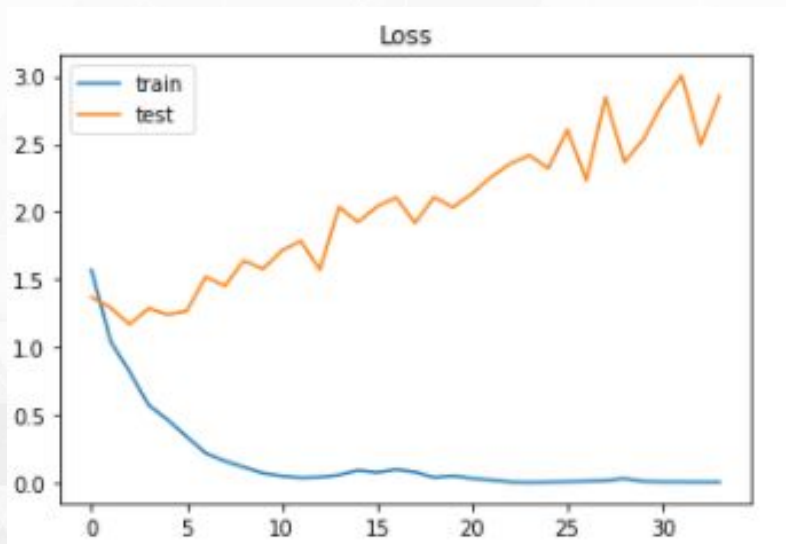
carcinoma_in_situ	0.17	0.20	0.18	30
light_dysplastic	0.21	0.19	0.20	37
moderate_dysplastic	0.07	0.07	0.07	30
normal_columnar	0.00	0.00	0.00	20
normal_intermediate	0.19	0.21	0.20	14
normal_superficial	0.09	0.07	0.08	15
severe_dysplastic	0.28	0.33	0.30	40
accuracy			0.17	186
macro avg	0.14	0.15	0.15	186
weighted avg	0.16	0.17	0.17	186





**batch: 16, dropout: 0.1, learning\_rate: 0.001**

Train: 0.999, Test: 0.568



# batch: 16, dropout: 0.1, learning\_rate: 0.001

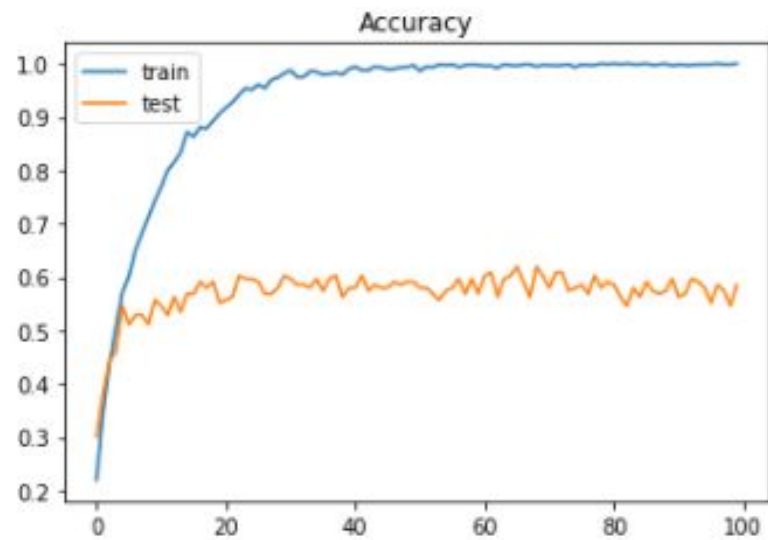
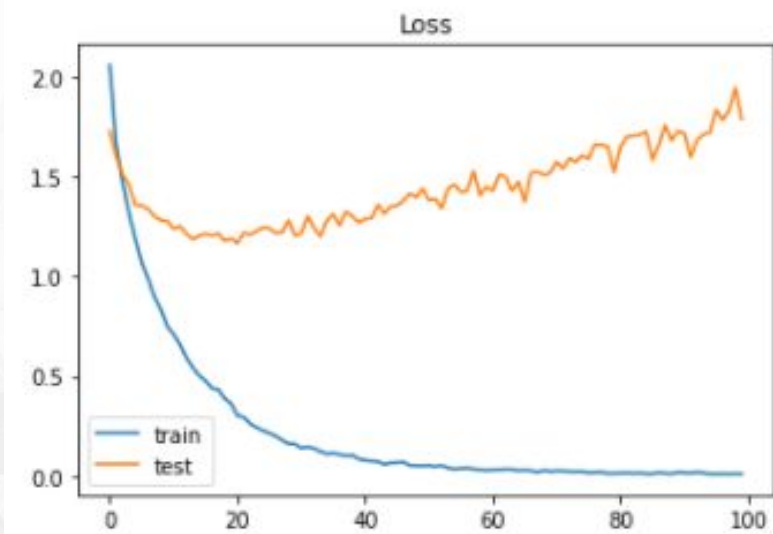
Train: 0.999, Test: 0.568

-----CLASSIFICATION-----				
	precision	recall	f1-score	support
carcinoma_in_situ	0.18	0.17	0.17	30
light_dysplastic	0.19	0.14	0.16	37
moderate_dysplastic	0.13	0.23	0.17	30
normal_columnar	0.07	0.05	0.06	20
normal_intermediate	0.15	0.14	0.15	14
normal_superficial	0.07	0.07	0.07	15
severe_dysplastic	0.19	0.17	0.18	40
accuracy			0.15	186
macro avg	0.14	0.14	0.14	186
weighted avg	0.15	0.15	0.15	186



**batch: 16, dropout: 0.2, learning\_rate: 0.0001**

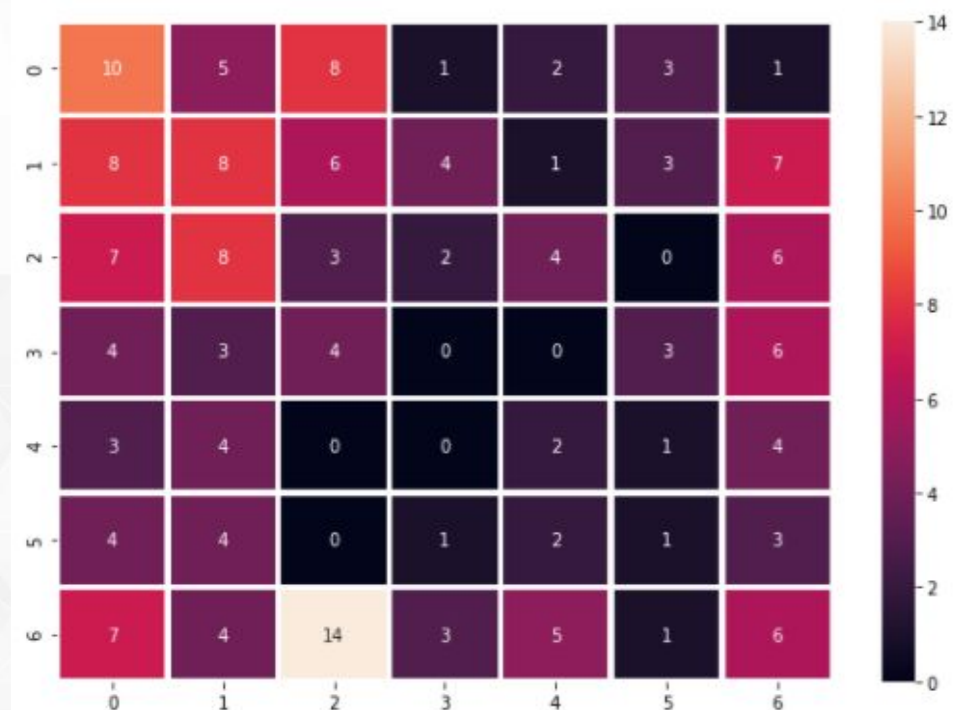
Train: 1.000, Test: 0.574



**batch: 16, dropout: 0.2, learning\_rate: 0.0001**

Train: 1.000, Test: 0.574

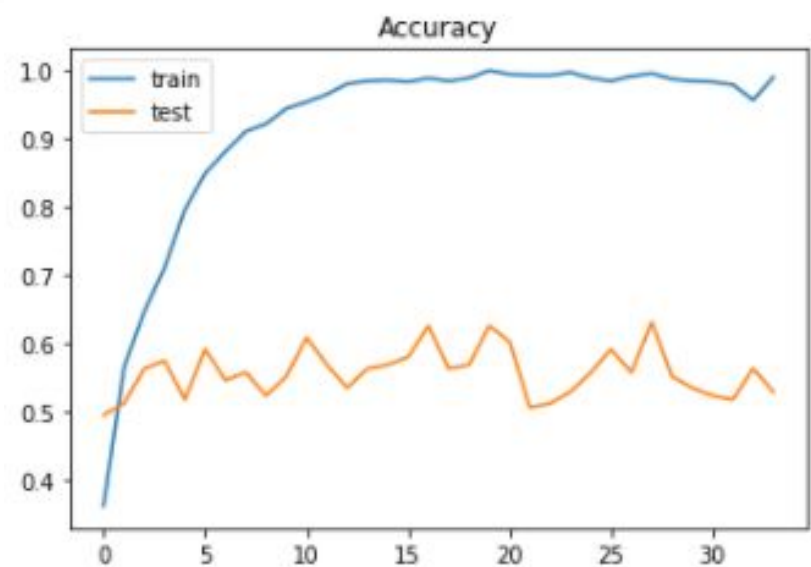
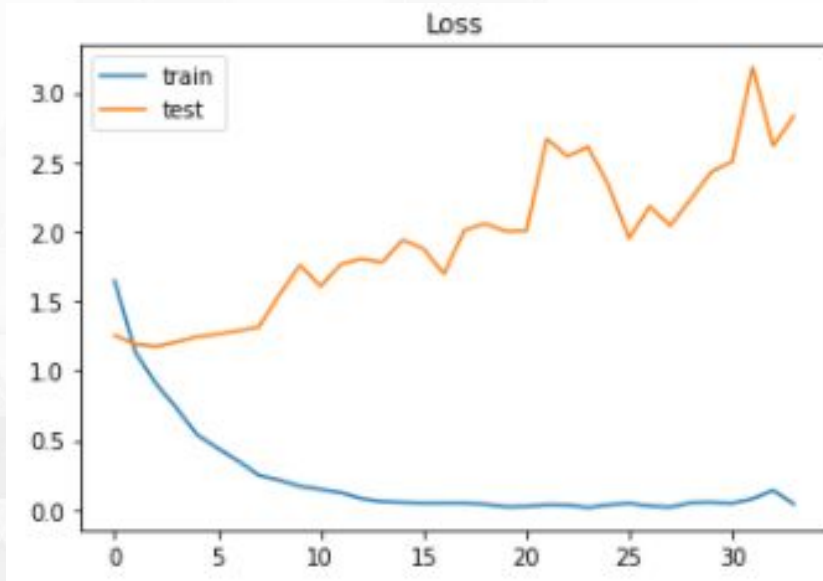
-----CLASSIFICATION-----				
	precision	recall	f1-score	support
carcinoma_in_situ	0.23	0.33	0.27	30
light_dysplastic	0.22	0.22	0.22	37
moderate_dysplastic	0.09	0.10	0.09	30
normal_columnar	0.00	0.00	0.00	20
normal_intermediate	0.12	0.14	0.13	14
normal_superficial	0.08	0.07	0.07	15
severe_dysplastic	0.18	0.15	0.16	40
accuracy			0.16	186
macro avg	0.13	0.14	0.14	186
weighted avg	0.15	0.16	0.15	186





**batch: 16, dropout: 0.2, learning\_rate: 0.001**

Train: 0.988, Test: 0.540

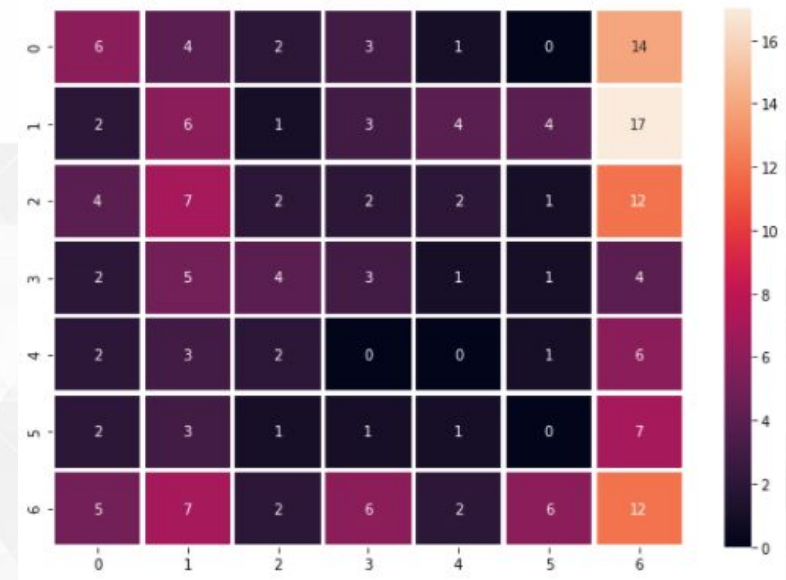




**batch: 16, dropout: 0.2, learning\_rate: 0.001**

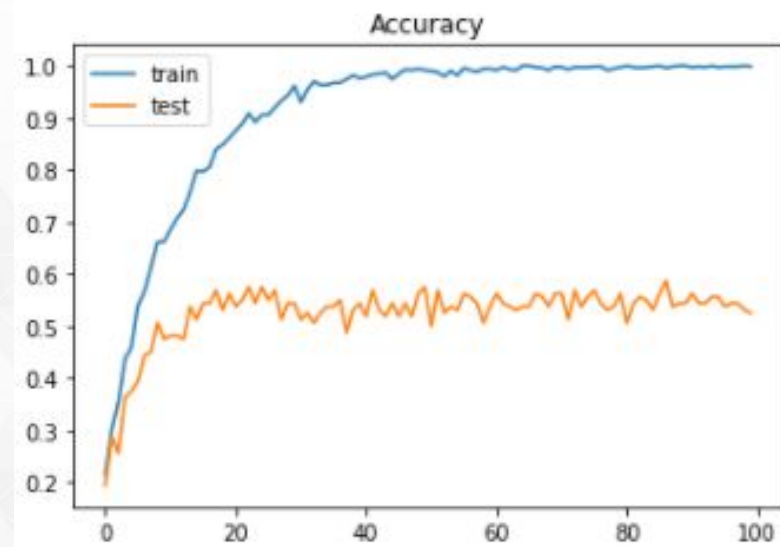
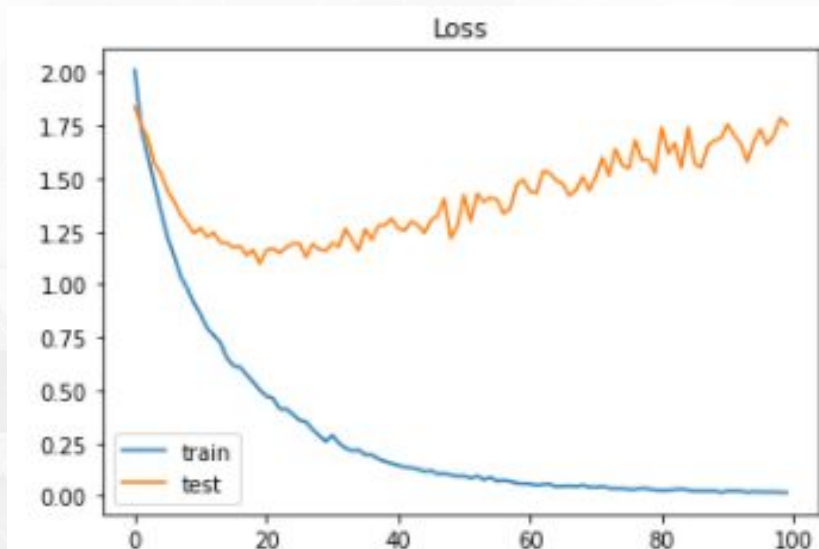
Train: 0.988, Test: 0.540

-----CLASSIFICATION-----				
	precision	recall	f1-score	support
carcinoma_in_situ	0.26	0.20	0.23	30
light_dysplastic	0.17	0.16	0.17	37
moderate_dysplastic	0.14	0.07	0.09	30
normal_columnar	0.17	0.15	0.16	20
normal_intermediate	0.00	0.00	0.00	14
normal_superficial	0.00	0.00	0.00	15
severe_dysplastic	0.17	0.30	0.21	40
accuracy			0.16	186
macro avg	0.13	0.13	0.12	186
weighted avg	0.15	0.16	0.15	186



**batch: 32, dropout: 0.1, learning\_rate: 0.0001**

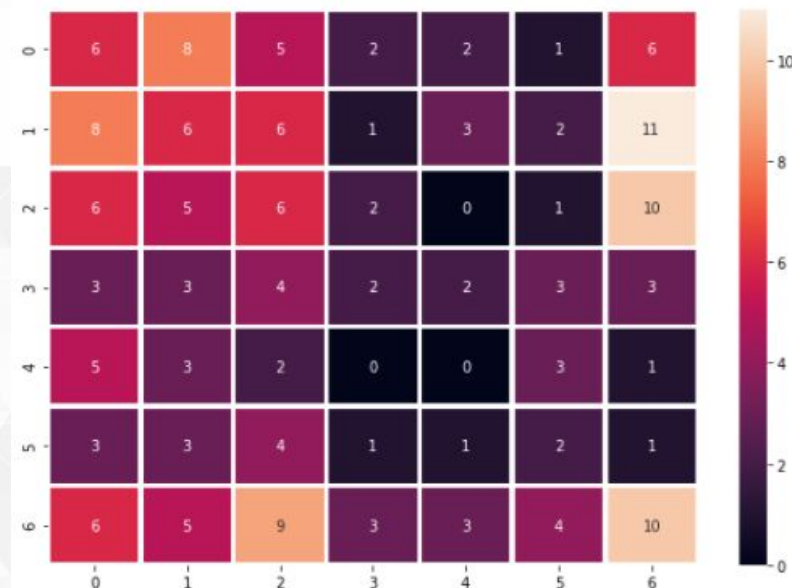
Train: 1.000, Test: 0.575



**batch: 32, dropout: 0.1, learning\_rate: 0.0001**

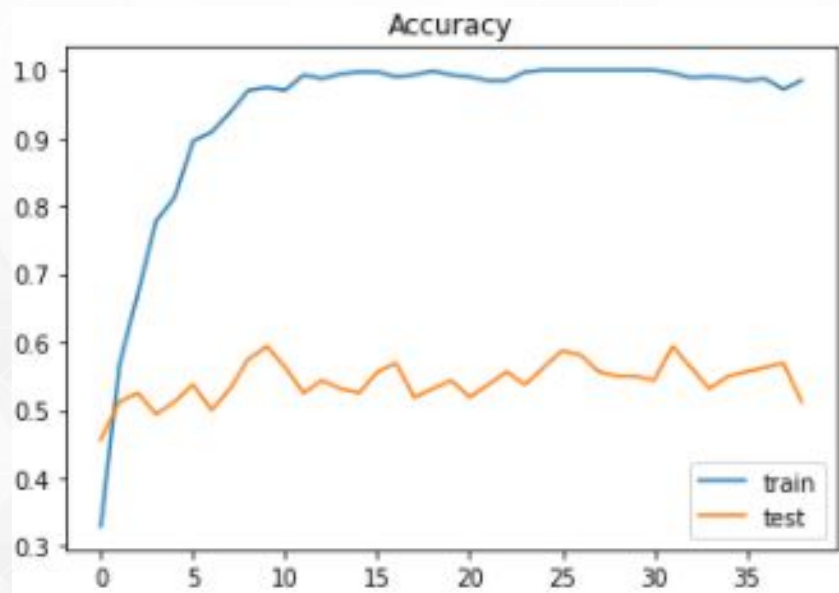
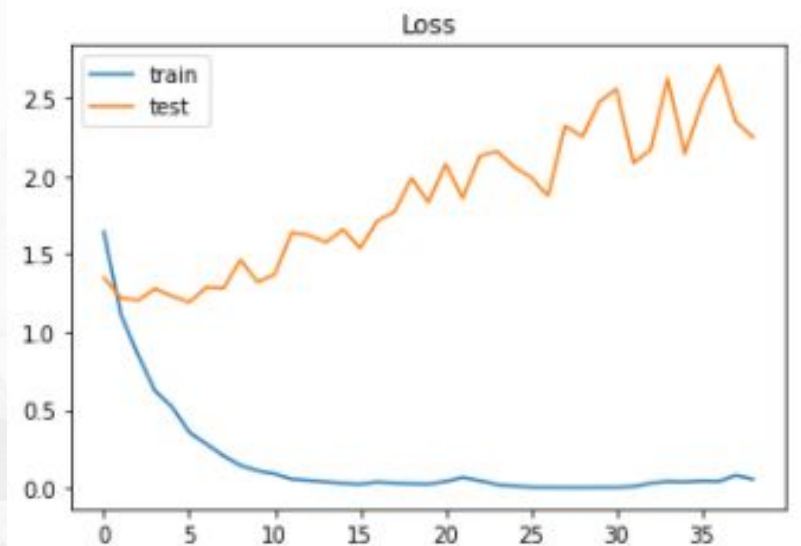
**Train: 1.000, Test: 0.575**

-----CLASSIFICATION-----				
	precision	recall	f1-score	support
carcinoma_in_situ	0.16	0.20	0.18	30
light_dysplastic	0.18	0.16	0.17	37
moderate_dysplastic	0.17	0.20	0.18	30
normal_columnar	0.18	0.10	0.13	20
normal_intermediate	0.00	0.00	0.00	14
normal_superficial	0.12	0.13	0.13	15
severe_dysplastic	0.24	0.25	0.24	40
accuracy			0.17	186
macro avg	0.15	0.15	0.15	186
weighted avg	0.17	0.17	0.17	186



**batch: 32, dropout: 0.1, learning\_rate: 0.001**

Train: 0.990, Test: 0.500





**batch: 32, dropout: 0.1, learning\_rate: 0.001**

**Train: 0.990, Test: 0.500**

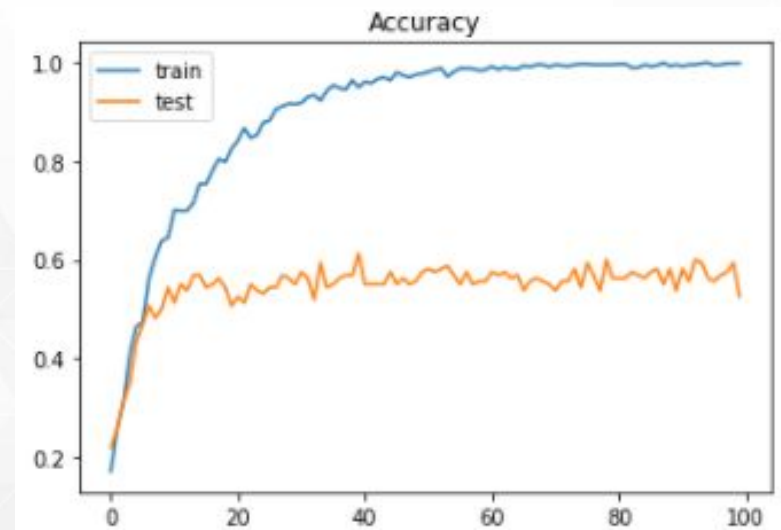
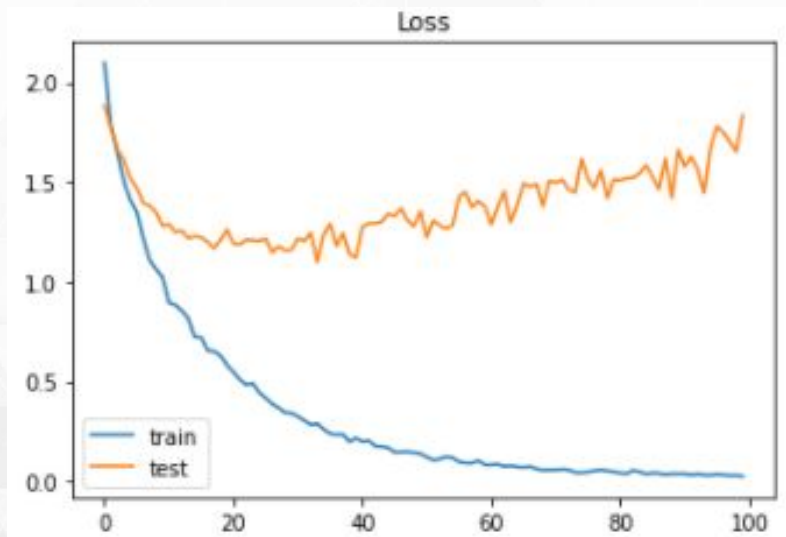
-----CLASSIFICATION-----				
	precision	recall	f1-score	support
carcinoma_in_situ	0.15	0.13	0.14	30
light_dysplastic	0.24	0.16	0.19	37
moderate_dysplastic	0.14	0.20	0.16	30
normal_columnar	0.20	0.05	0.08	20
normal_intermediate	0.19	0.21	0.20	14
normal_superficial	0.14	0.13	0.14	15
severe_dysplastic	0.24	0.33	0.27	40
accuracy			0.19	186
macro avg	0.18	0.17	0.17	186
weighted avg	0.19	0.19	0.18	186





**batch: 32, dropout: 0.1, learning\_rate: 0.0001**

Train: 1.000, Test: 0.587



**batch: 32, dropout: 0.1, learning\_rate: 0.0001**

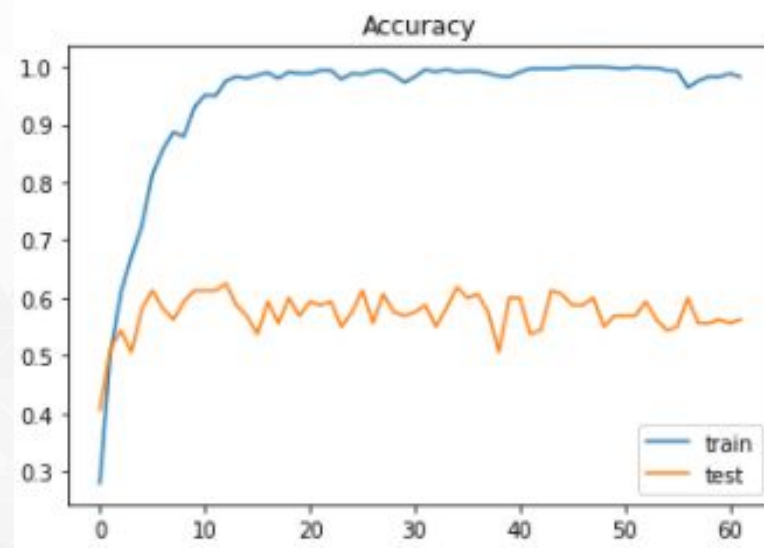
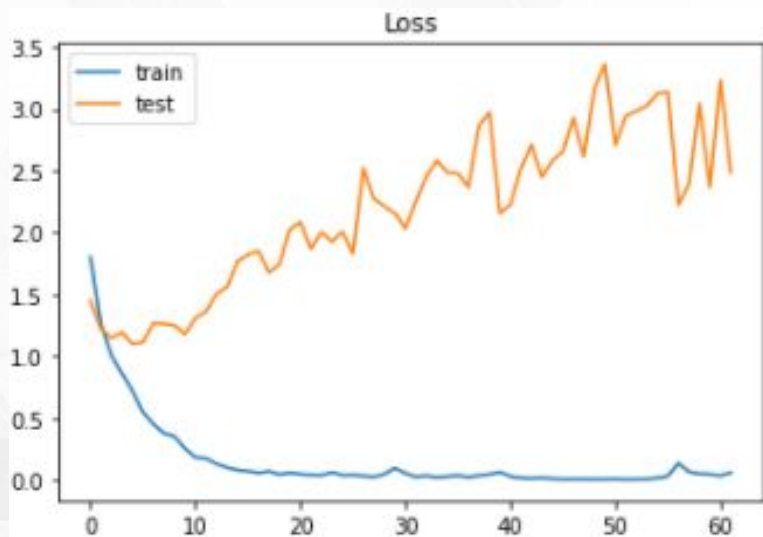
Train: 1.000, Test: 0.587

-----CLASSIFICATION-----				
	precision	recall	f1-score	support
carcinoma_in_situ	0.18	0.20	0.19	30
light_dysplastic	0.29	0.27	0.28	37
moderate_dysplastic	0.19	0.17	0.18	30
normal_columnar	0.14	0.15	0.15	20
normal_intermediate	0.07	0.07	0.07	14
normal_superficial	0.17	0.13	0.15	15
severe_dysplastic	0.22	0.25	0.24	40
accuracy			0.20	186
macro avg	0.18	0.18	0.18	186
weighted avg	0.20	0.20	0.20	186



**batch: 32, dropout: 0.1, learning\_rate: 0.001**

Train: 0.997, Test: 0.562



**batch: 32, dropout: 0.1, learning\_rate: 0.001**

Train: 0.997, Test: 0.562

-----CLASSIFICATION-----				
	precision	recall	f1-score	support
carcinoma_in_situ	0.17	0.17	0.17	30
light_dysplastic	0.24	0.22	0.23	37
moderate_dysplastic	0.18	0.13	0.15	30
normal_columnar	0.08	0.10	0.09	20
normal_intermediate	0.00	0.00	0.00	14
normal_superficial	0.08	0.07	0.07	15
severe_dysplastic	0.20	0.25	0.22	40
accuracy			0.16	186
macro avg	0.14	0.13	0.13	186
weighted avg	0.16	0.16	0.16	186





# Resultados e Discussões



## Resultados

100epochs\_32batch\_0.2dropout\_0.001learningrate\_128x128 - **Train: 0.997, Test: 0.562**  
100epochs\_32batch\_0.2dropout\_0.0001learningrate\_128x128 - **Train: 1.000, Test: 0.587**

100epochs\_16batch\_0.2dropout\_0.001learningrate\_128x128 - **Train: 0.988, Test: 0.540**  
100epochs\_16batch\_0.2dropout\_0.0001learningrate\_128x128 - **Train: 1.000, Test: 0.574**

100epochs\_32batch\_0.1dropout\_0.001learningrate\_128x128 - **Train: 0.990, Test: 0.500**  
100epochs\_32batch\_0.1dropout\_0.0001learningrate\_128x128 - **Train: 1.000, Test: 0.575**

100epochs\_16batch\_0.1dropout\_0.001learningrate\_128x128 - **Train: 0.999, Test: 0.568**  
100epochs\_16batch\_0.1dropout\_0.0001learningrate\_128x128 - **Train: 1.000, Test: 0.591**

# Conclusões

## Conclusões

**Overfitting:** ocorre quando nosso modelo se torna realmente bom em ser capaz de classificar ou prever dados que foram incluídos no conjunto de treinamento, mas não é tão bom em classificar dados nos quais não foi treinado. Então, essencialmente, o modelo super ajustou os dados no conjunto de treinamento.

# Referências

## Referências

- <https://www.sciencedirect.com/science/article/pii/S2352914820302537#bib33>
- <https://keras.io/api/applications/resnet/>
- <https://cv-tricks.com/keras/understand-implement-resnets/>
- <https://www.machinecurve.com/index.php/2020/01/30/what-are-max-pooling-average-pooling-global-max-pooling-and-global-average-pooling/>
- <https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/#:~:text=The%20rectified%20linear%20activation%20function,otherwise%2C%20it%20will%20output%20zero.>
- <https://machinelearningmastery.com/softmax-activation-function-with-python/>
- <https://deeplizard.com/learn/video/DEMmkFC6lGM#:~:text=Overfitting%20occurs%20when%20our%20model,data%20in%20the%20training%20set.>