

Conclusões

Resultados

Implementação

CNN

Redes Neurais

Contagem e identificação  
de leucócitos



**INSTITUTO POLITÉCNICO DE BRAGANÇA**  
Escola Superior de Tecnologia e de Gestão

## **Obtenção de Modelos de Deep learning para a Classificação Automática de Leucócitos**

Romeu Beato, 7943

Apresentação da Dissertação de Mestrado em Tecnologia Biomédica  
Orientação: Pedro João Soares Rodrigues (PhD)

Bragança, 7 de Dezembro de 2018

Conclusões

Resultados

Implementação

CNN

Redes Neurais

Contagem e identificação  
de leucócitos

## OBJETIVOS DO TRABALHO

- I. Fornecer uma **ferramenta computacional** para permitir a classificação de leucócitos de forma automática a partir de imagens obtidas em trabalhos de microscopia
- II. Comparar as **performances de redes** vencedoras do concurso ILSVRC na **classificação de leucócitos**.
- III. Comparar a performance na identificação de leucócitos após realização do treino das **redes treinadas de raiz e com *Transfer Learning***.

Conclusões

Resultados

Implementação

CNN

Redes Neurais

## CONTAGEM E IDENTIFICAÇÃO DE LEUCÓCITOS – TIME LINE



**1855**

**CRAMER**

CONTAGEM EM ESPAÇO  
CAPILAR



**1869**

**POTAIN**

PIPETA DE DILUIÇÃO



**~1869**

**HAVEM**

LÂMINA DE ESPESSURA  
PRECISA E MIRÓMETRO  
OCULAR



**1877**

**GOWER**

MELHORA A CÂMARA DE  
CONTAGEM DE HAVEM,  
COLOCANDO DIVISÕES



**~1880**

**THOMA**

MELHORA A CÂMARA DE  
CONTAGEM



**~1880**

**MALASSEZ**

INTRODUZ UM  
HEMATÓCRITO COM  
PROPRIEDADES  
MICROFOTOGRAFICAS



**1896**

HEMATÓCRITO COM BASE  
NO PRINCÍPIO  
TURBIDOMÉTRICO  
(PROPRIEDADES ÓTICAS À  
LUZ DA VELA)



**1903**

**STRONG & SELIGMAN**

CONTAGEM COM  
DILUIÇÃO DE 5 ML DE  
SANGUE NA PROPORÇÃO  
1:100 EM VIOLETA DE  
METILO.



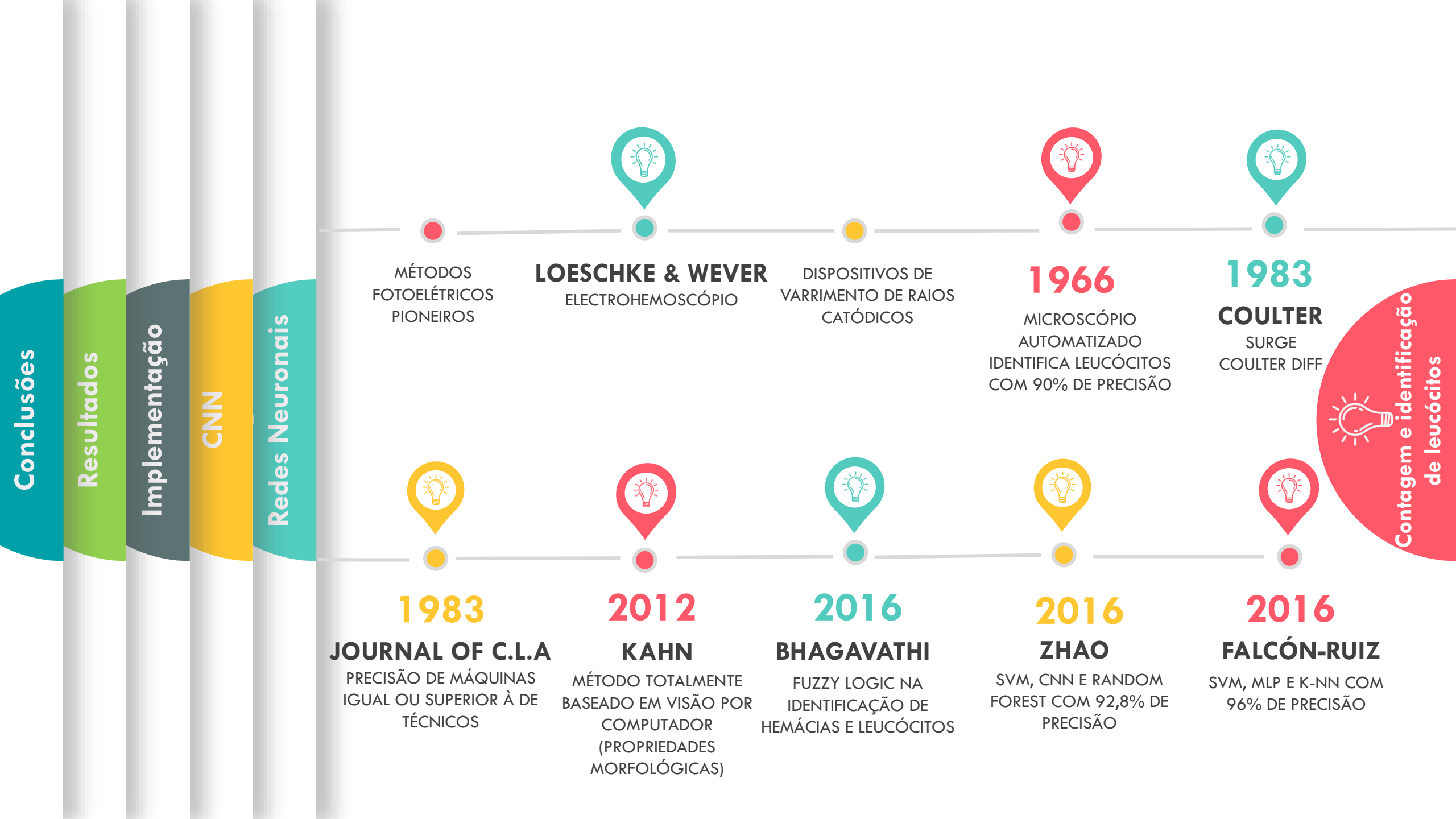
**~1903**

**BURKER**

INVENTA CÂMARA  
PASSÍVEL DE ENCHER POR  
CAPILARIDADE APÓS  
COLOCADA A  
COBERTURA



Contagem e identificação  
de leucócitos



Conclusões

Resultados

Implementação

CNN

Redes Neurais

MÉTODOS  
FOTOELÉTRICOS  
PIONEIROS

LOESCHKE & WEVER  
ELECTROHEMOSCÓPIO

DISPOSITIVOS DE  
VARRIMENTO DE RAIOS  
CATÓDICOS

1966  
MICROSCÓPIO  
AUTOMATIZADO  
IDENTIFICA LEUCÓCITOS  
COM 90% DE PRECISÃO

1983  
COULTER  
SURGE  
COULTER DIFF

1983  
JOURNAL OF C.L.A  
PRECISÃO DE MÁQUINAS  
IGUAL OU SUPERIOR À DE  
TÉCNICOS

2012  
KAHN  
MÉTODO TOTALMENTE  
BASEADO EM VISÃO POR  
COMPUTADOR  
(PROPRIEDADES  
MORFOLÓGICAS)

2016  
BHAGAVATHI  
FUZZY LOGIC NA  
IDENTIFICAÇÃO DE  
HEMÁCIAS E LEUCÓCITOS

2016  
ZHAO  
SVM, CNN E RANDOM  
FOREST COM 92,8% DE  
PRECISÃO

2016  
FALCÓN-RUIZ  
SVM, MLP E K-NN COM  
96% DE PRECISÃO

Contagem e identificação  
de leucócitos

## REDES NEURONAIS

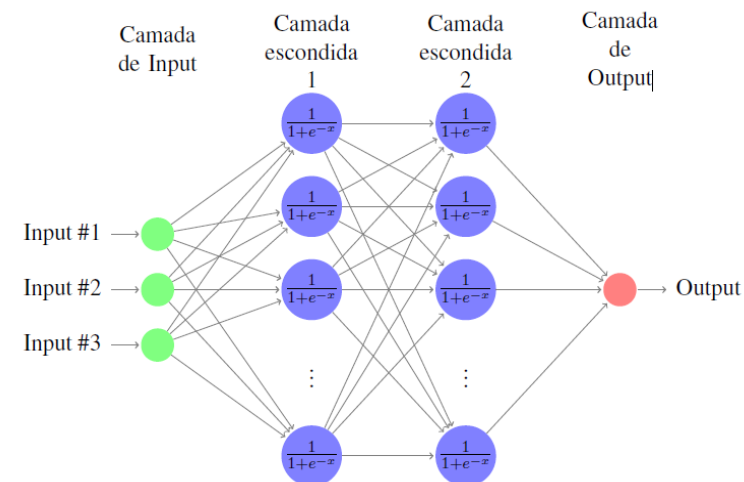
A origem do **Deep Learning** reporta à década de 80. Consiste num **sistema de computação paralela** com um grande número de **processadores interligados, organizados por camadas**.

O processo de aprendizagem pode ser visto como um **problema de atualização da arquitetura da rede e dos pesos das ligações**.

De entre os **paradigmas básicos de aprendizagem** máquina destacam-se:

- Supervisionada
- Não supervisionada
- De reforço.

O *deep learning*, que facilita a utilização de **camadas escondidas**, facilita uma representação mais complexa de padrões nos dados.



Redes Neurais

Contagem e identificação  
de leucócitos

Conclusões

Resultados

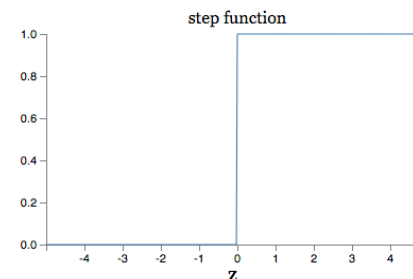
Implementação

CNN

## FUNÇÃO STEP E O PERCEPTRÃO



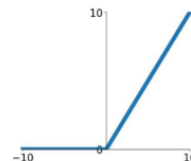
O output de um neurónio, 0 ou 1, é determinado pelo facto de a soma ponderada  $\sum_j w_j x_j$  ser maior ou menor do que um valor limiar (ou *threshold*).



## FUNÇÕES DE ATIVAÇÃO

**ReLU**

$$\max(0, x)$$



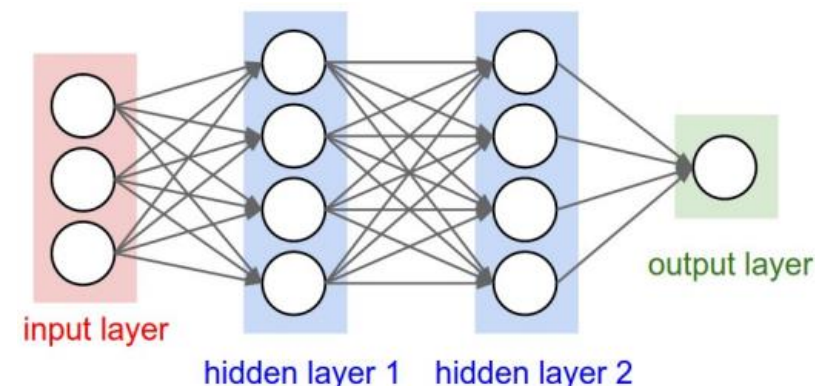
**Softmax**

$$\phi_i = \frac{e^{z_i}}{\sum_{j \in \text{group}} e^{z_j}}$$

## FEEDFORWARD



As redes em que o *output* de um neurónio é utilizado como *input* do neurónio da camada seguinte são chamadas de redes neuronais **feedforward**.

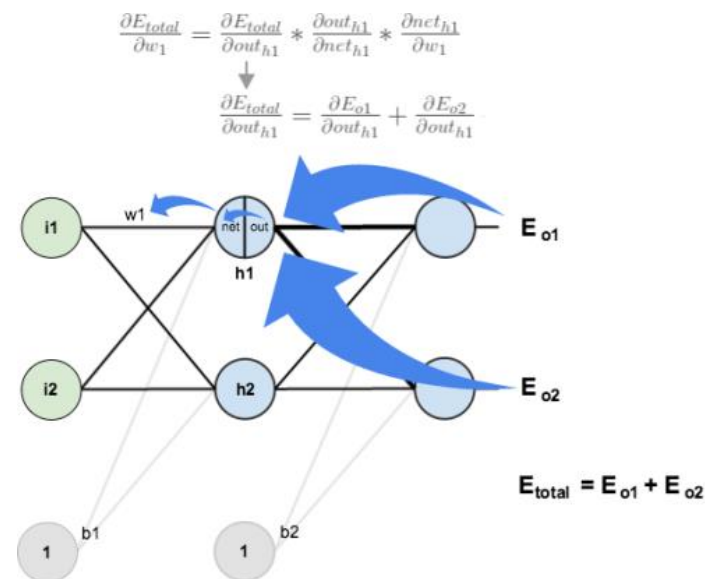


## BACKPROPAGATION

📍 Durante o treino, a performance da rede neuronal é avaliada pelo cálculo da diferença entre os seus *outputs* e o *output* desejado para todos os exemplos de treino.

$$E = \frac{1}{2} \sum_{k \in T_r} \sum_{j=1}^m (y_j(x_k, w) - d_{jk})^2$$

• onde  $d_{jk}$  é o elemento  $j$  de  $d_k$ ,  
 $y_j(x_k, w)$  é o *output*  $j$  da rede neuronal para o *input*  $x_k$ , e  $T_r$  é um índice do conjunto de treino.



Fonte: <https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/>

📍 Os pesos  $w$  são ajustados durante o treino, de forma a que este erro seja minimizado.

## MÉTODOS DE OTIMIZAÇÃO

**Stochastic Gradient Descent (SGD)**

O SGD é calculado com alguns exemplos (batch) iterativamente (em vez de toda a base de treino).

$$\theta = \theta - \alpha * \sum_{k=i}^{i+m} \nabla_{\theta} J(\theta; x^{(k)}, y^{(k)})$$

onde  $\theta$  é o parâmetro a atualizar,  
 $\alpha$  é a taxa de aprendizagem e  
 $m$  é o tamanho do mini-lote (batch).

**RMSprop**

O *Root Mean Square Propagation* é um dos algoritmos de gradiente adaptativo mais usados para o treino de redes neurais profundas.

$$E[g^2]_t = 0,9 E[g^2]_{t-1} + 0,1 g_t^2$$

$$\theta_{t+1} = \theta_t - \frac{n}{\sqrt{E[g^2]_t + \varepsilon}} g_t$$

onde  $g$  é o gradiente num dado instante  $t$ ,  
 $\eta$  é o *learning rate*  
 $\theta$  representa os parâmetros a atualizar ( $W$  e  $b$ ).

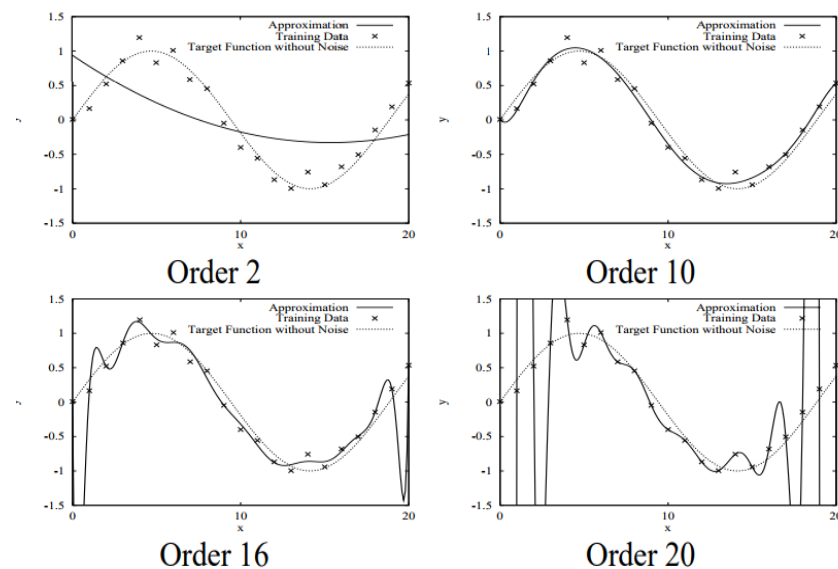






**Overfitting** traduz-se numa sobre-adaptação de um determinado modelo a um conjunto de dados, perdendo este modelo a capacidade de generalização.

## OVERFITTING



Adaptação de modelos polinomiais à equação  $y = \sin(x/3) + v$ .

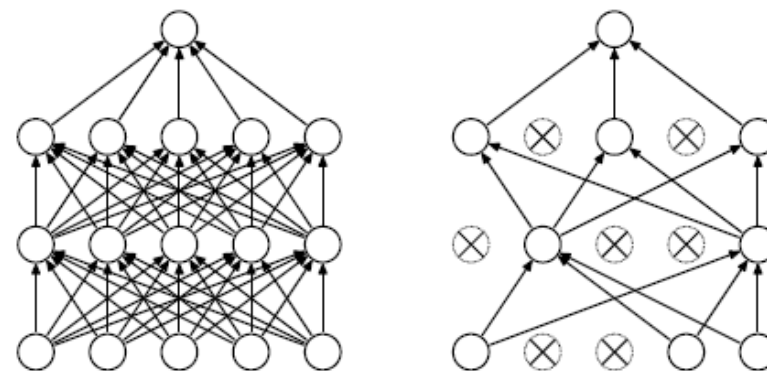
Fonte

<https://www.semanticscholar.org/paper/Lessons-in-Neural-Network-Training/%3A-Overfitting-May-Lawrence-Giles/3e0a8efc5255abc4f92f8e5f1db257f8acfa8233/figure/0>



**Dropout** é uma técnica utilizada para evitar o **overfitting**. O termo **dropout** refere-se à exclusão de unidades (escondidas e visíveis) numa rede neuronal.

## DROPOUT



Fonte: <https://medium.com/@amarbudhiraja/https-medium-com-amarbudhiraja-learning-less-to-learn-better-dropout-in-deep-machine-learning-74334da4bfc5>



## BATCH NORMALIZATION



Esta normalização em lote adiciona dois parâmetros treináveis a cada camada e lote, permitindo que o SGD (ou outro otimizador) faça a desnormalização alterando apenas esses dois pesos para cada ativação, em vez de perder a estabilidade da rede, alterando todos os pesos.

## DATA AUGMENTATION



O ***data augmentation*** aumenta a diversidade de dados usando informação existente apenas no conjunto de dados disponível. São realizadas transformações ao nível da sua cor e geometria, como a reflexão, recorte, mudança na paleta de cores e tradução da imagem.



Exemplos de transformações tradicionais, realizadas numa imagem. Fonte:[Perez and Wang, 2017].

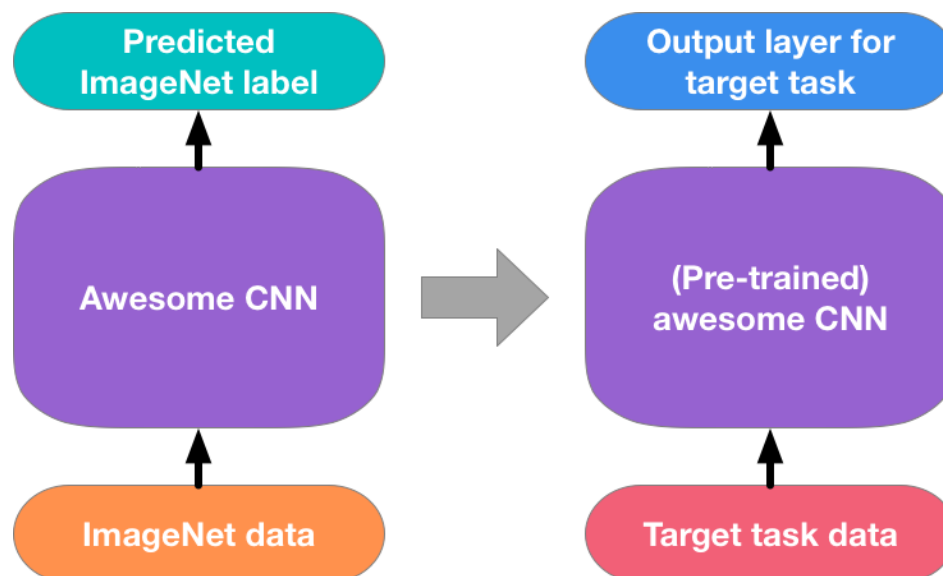


## FINE-TUNNING



O *fine-tuning* é um procedimento baseado no conceito de transferência de aprendizagem (*transfer learning*).

Começa-se por treinar uma CNN para aprender características para um domínio amplo com uma função de classificação voltada para minimizar o erro nesse domínio. Em seguida, substitui-se a função de classificação e otimiza-se a rede novamente para minimizar o erro noutro domínio mais específico. Sob esta configuração, é feita a transferência de características e parâmetros da rede do domínio amplo para outro mais específico [Reyes et al., 2015].

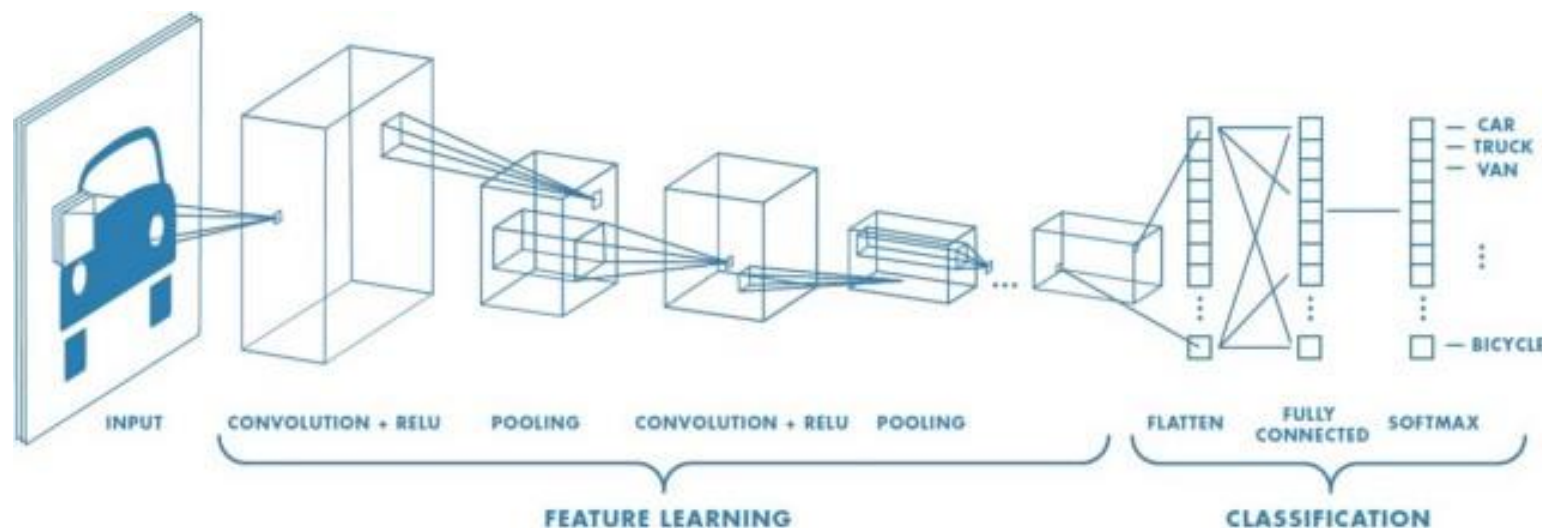


Fonte: [https://gluon.mxnet.io/chapter08\\_computer-vision/fine-tuning.html](https://gluon.mxnet.io/chapter08_computer-vision/fine-tuning.html)



## REDES NEURONAIS CONVOLUCIONAIS

- As **redes neuronais convolucionais** são úteis na classificação, segmentação e detecção de objetos em imagens [Wu, 2017].
- Ao passar por mais camadas de convolução, obtém-se mapas de ativação que representam recursos cada vez mais complexos [Deshpande, 2016].



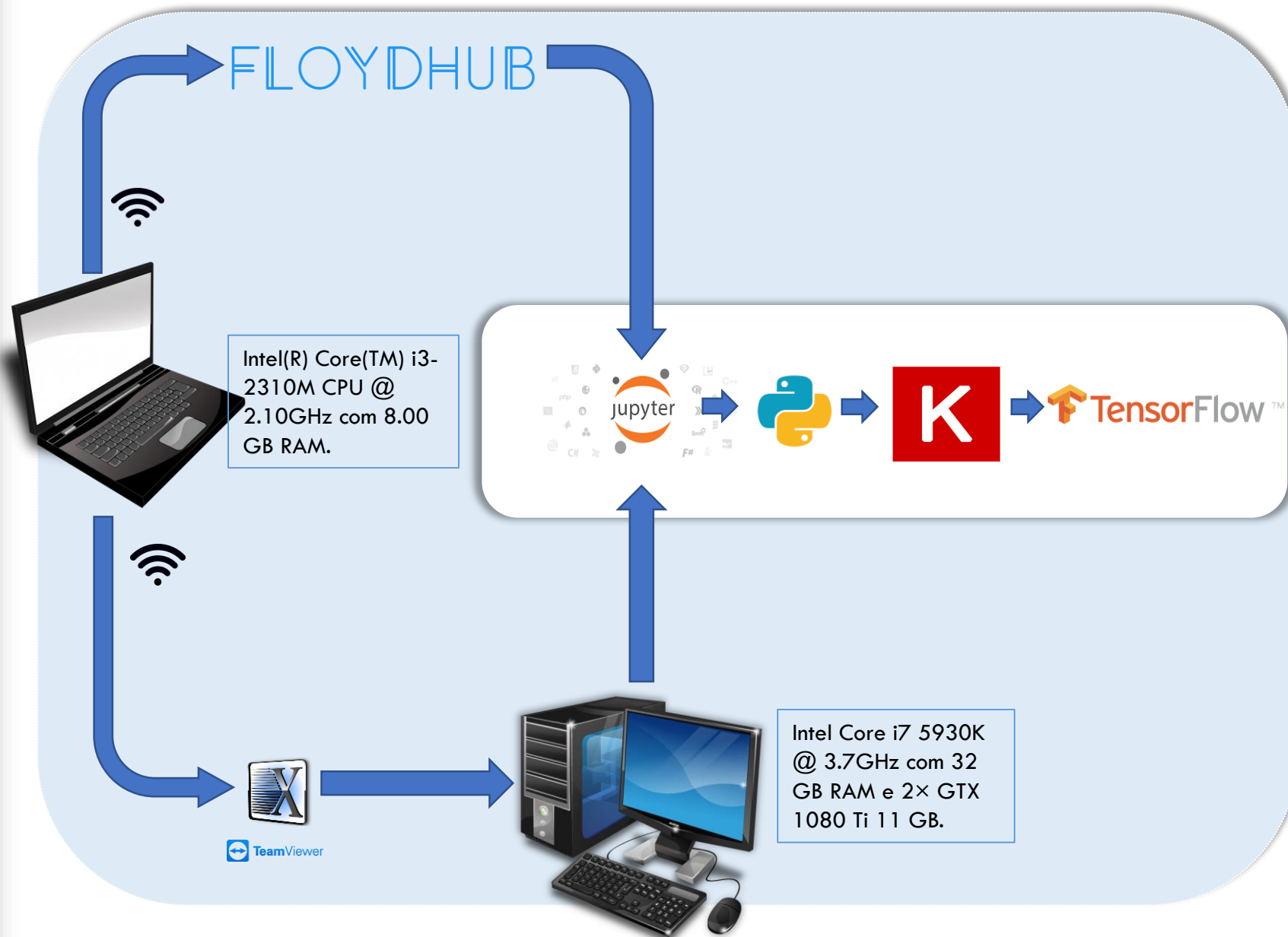
Fonte: <https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148>

- A perda de informação espacial é compensada por um número crescente de mapas de recursos nas camadas mais altas [Scherer et al., 2010].



CNN

## SETUP UTILIZADO



Implementação

CNN

Redes Neurais

Contagem e identificação  
de leucócitos

# PRÉ-PROCESSAMENTO DE DADOS

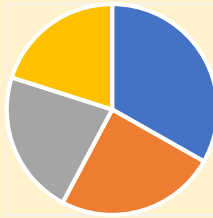


598 Leucócitos

Proveniência das imagens: GitHub, Google...

DISTRIBUIÇÃO APROXIMADA DAS CATEGORIAS DE LEUCÓCITOS

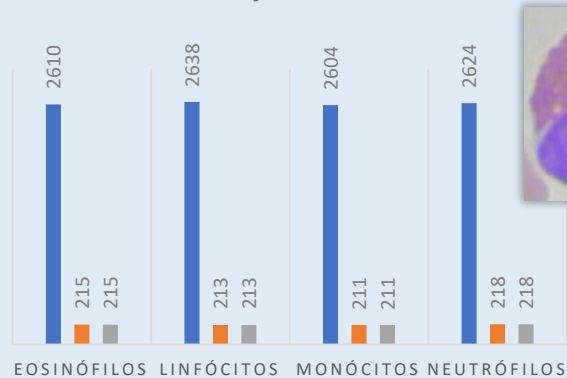
- Neurófilos
- Linfócitos
- Eosinófilos
- Monócitos



Categorização manual  
+  
GIMP2

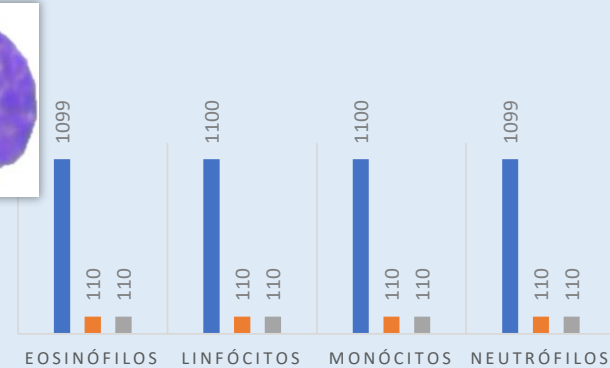
(Re)categorização  
+  
PAINT3D

QUANTIDADE DE IMAGENS NOS DATASETS DE TREINO, VALIDAÇÃO E TESTE APÓS A SEGMENTAÇÃO COM O GIMP2



■ Treino ■ Validação ■ Teste

QUANTIDADE DE IMAGENS NOS DATASETS DE TREINO, VALIDAÇÃO E TESTE APÓS A SEGMENTAÇÃO COM O PAINT3D



■ Treino ■ Validação ■ Teste



Implementação

CNN

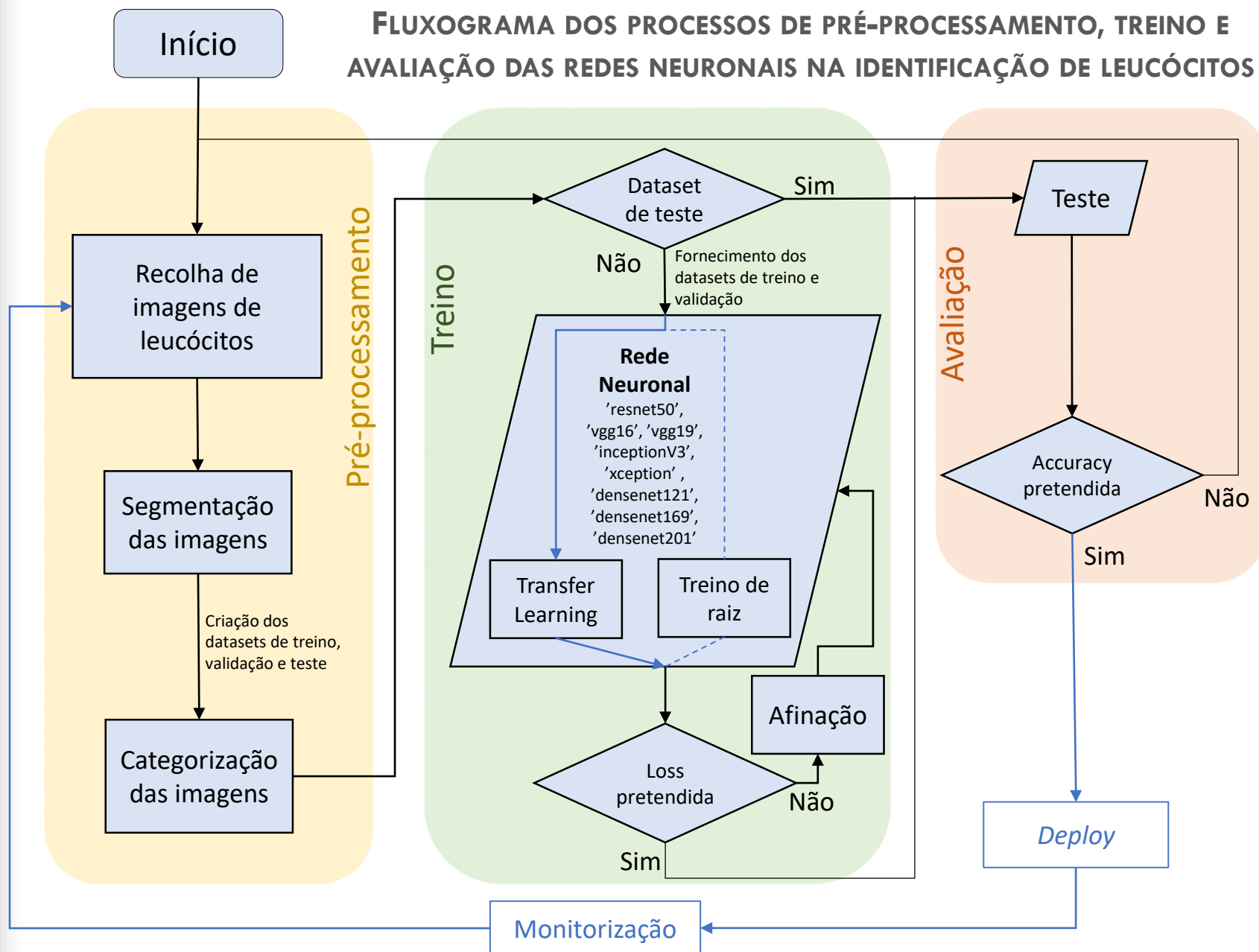
Redes Neurais

Contagem e identificação  
de leucócitos

Conclusões

Resultados

# FLUXOGRAMA DOS PROCESSOS DE PRÉ-PROCESSAMENTO, TREINO E AVALIAÇÃO DAS REDES NEURONAIS NA IDENTIFICAÇÃO DE LEUCÓCITOS



Implementação

CNN

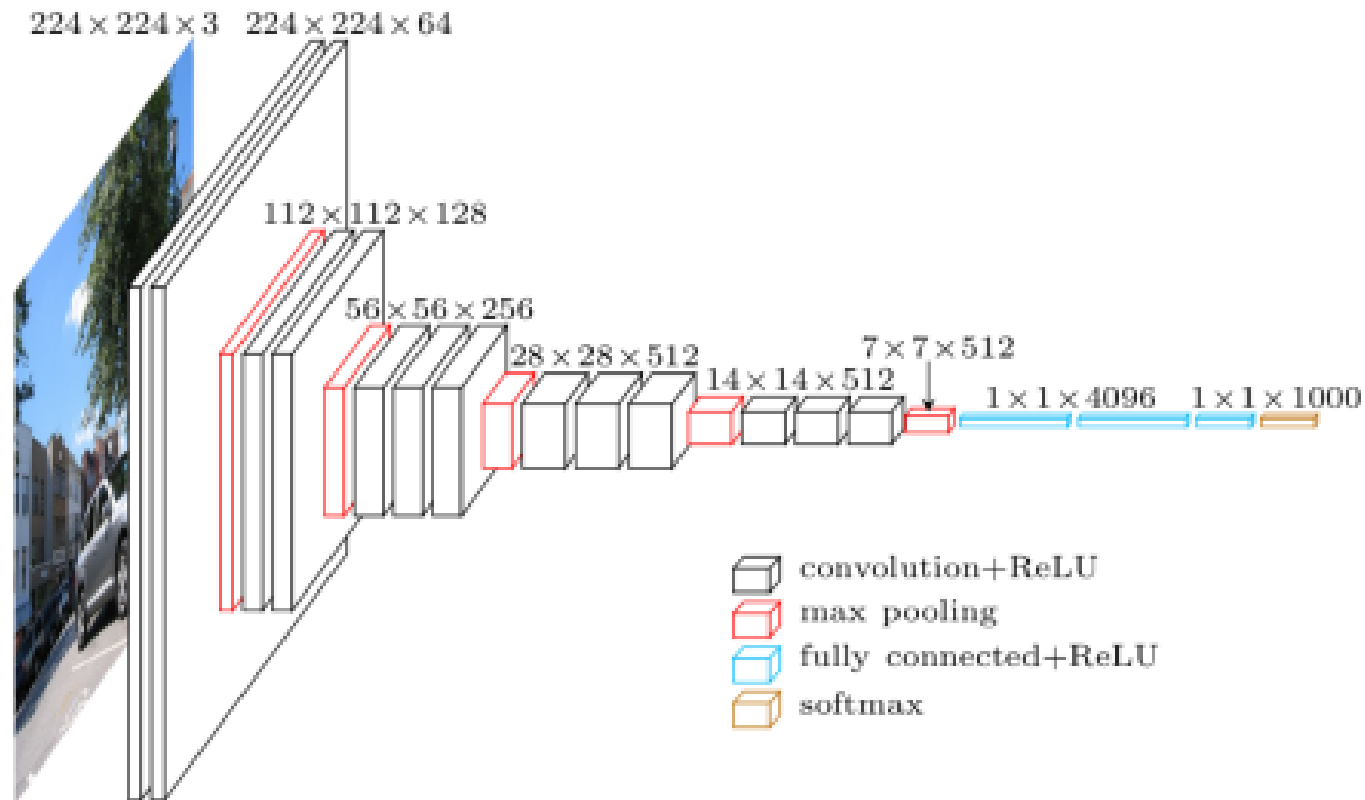
Redes Neurais

Contagem e identificação  
de leucócitos

## VGG16 E VGG19



A rede **VGG** utiliza apenas convoluções  $3 \times 3$ . A redução de volume de dados é conseguida através de *max pooling* [Simonyan and Zisserman, 2014].



Fonte: <https://www.cs.toronto.edu/~frossard/post/vgg16/>



Implementação

CNN

Redes Neurais

Contagem e identificação  
de leucócitos

Conclusões

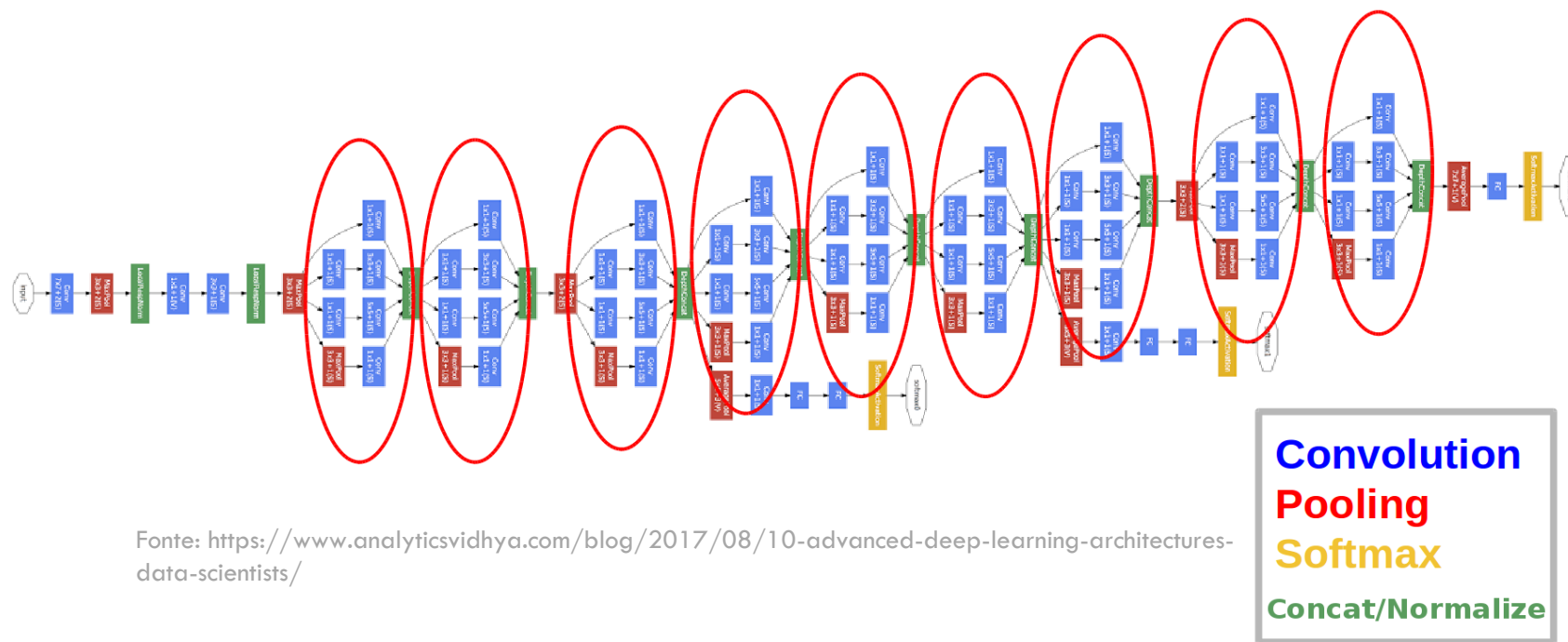
Resultados



## INCEPTION V3



A micro-arquitetura **Inception** actua como um extrator de *features* de vários níveis calculando várias convoluções no mesmo módulo [Rosenbrock, 2017].



## Implementação

## CNN

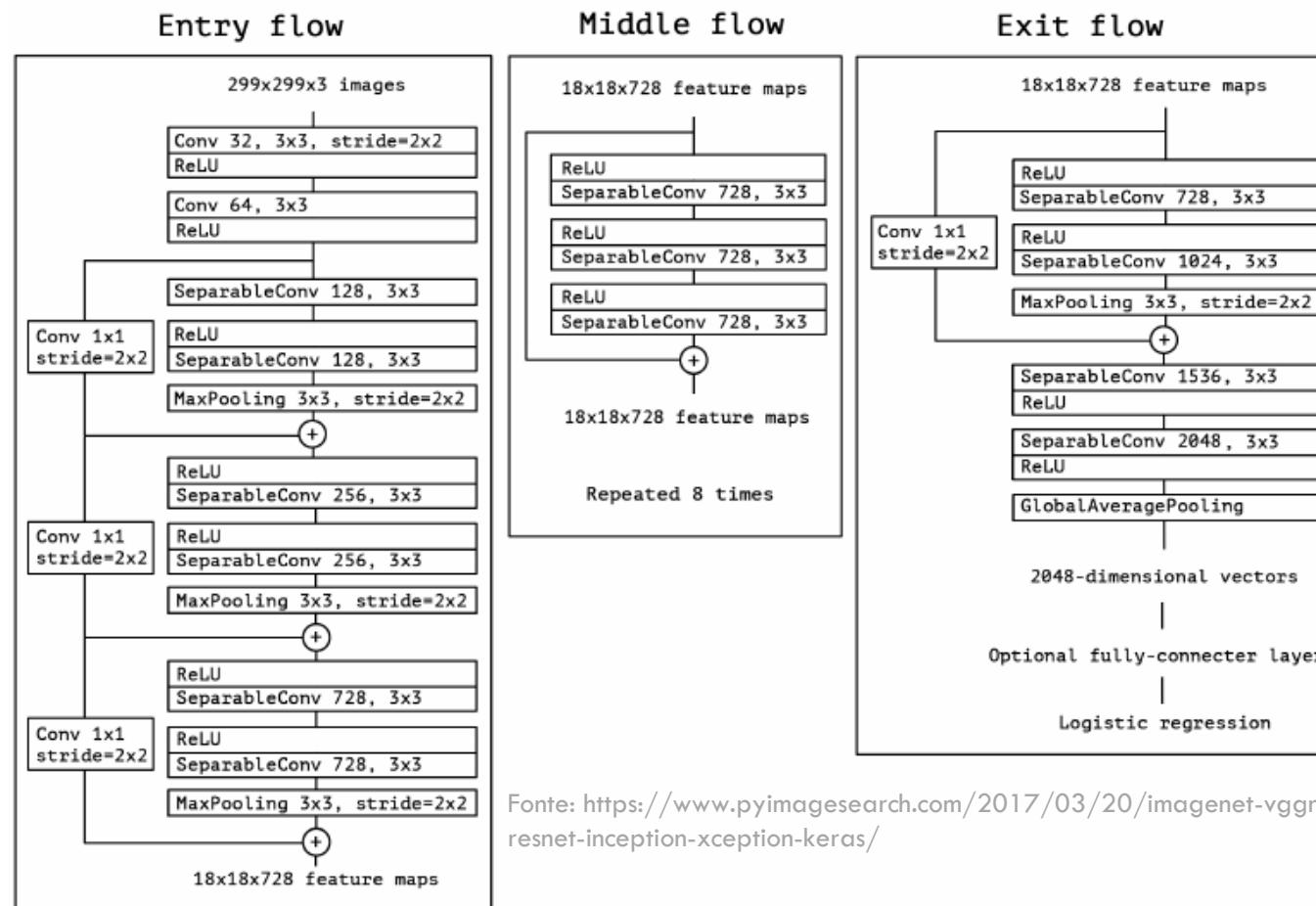
## Redes Neuronais

Contagem e identificação  
de leucócitos

## XCEPTION



A *Xception* foi proposta pelo criador do Keras, François Chollete, e é uma extensão da arquitetura *Inception* que substitui os módulos *Inception* por convoluções separáveis por profundidade.



Fonte: <https://www.pyimagesearch.com/2017/03/20/imagenet-vggnet-resnet-inception-xception-keras/>



Implementação

CNN

Redes Neurais

Contagem e identificação  
de leucócitos

## DENSENET 121, DENSENET 169 E DENSENET 201



A arquitetura **DenseNet** diferencia explicitamente entre informações adicionadas à rede e informações preservadas. As camadas são muito estreitas adicionando apenas um pequeno conjunto de mapas de recursos ao “conhecimento coletivo” da rede, mantendo inalterados os mapas de recursos restantes.

Layers	Output Size	DenseNet-121( $k = 32$ )	DenseNet-169( $k = 32$ )	DenseNet-201( $k = 32$ )
Convolution	$112 \times 112$	$7 \times 7$ conv, stride 2		
Pooling	$56 \times 56$	$3 \times 3$ max pool, stride 2		
Dense Block (1)	$56 \times 56$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$
Transition Layer (1)	$56 \times 56$	$1 \times 1$ conv		
	$28 \times 28$	$2 \times 2$ average pool, stride 2		
Dense Block (2)	$28 \times 28$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$
Transition Layer (2)	$28 \times 28$	$1 \times 1$ conv		
	$14 \times 14$	$2 \times 2$ average pool, stride 2		
Dense Block (3)	$14 \times 14$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 24$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 48$
Transition Layer (3)	$14 \times 14$	$1 \times 1$ conv		
	$7 \times 7$	$2 \times 2$ average pool, stride 2		
Dense Block (4)	$7 \times 7$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 16$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$
Classification Layer	$1 \times 1$	$7 \times 7$ global average pool		
		1000D fully-connected, softmax		

Fonte: <https://medium.com/@14prakash/understanding-and-implementing-architectures-of-resnet-and-resnext-for-state-of-the-art-image-cf51669e1624>



Implementação

CNN

Redes Neurais

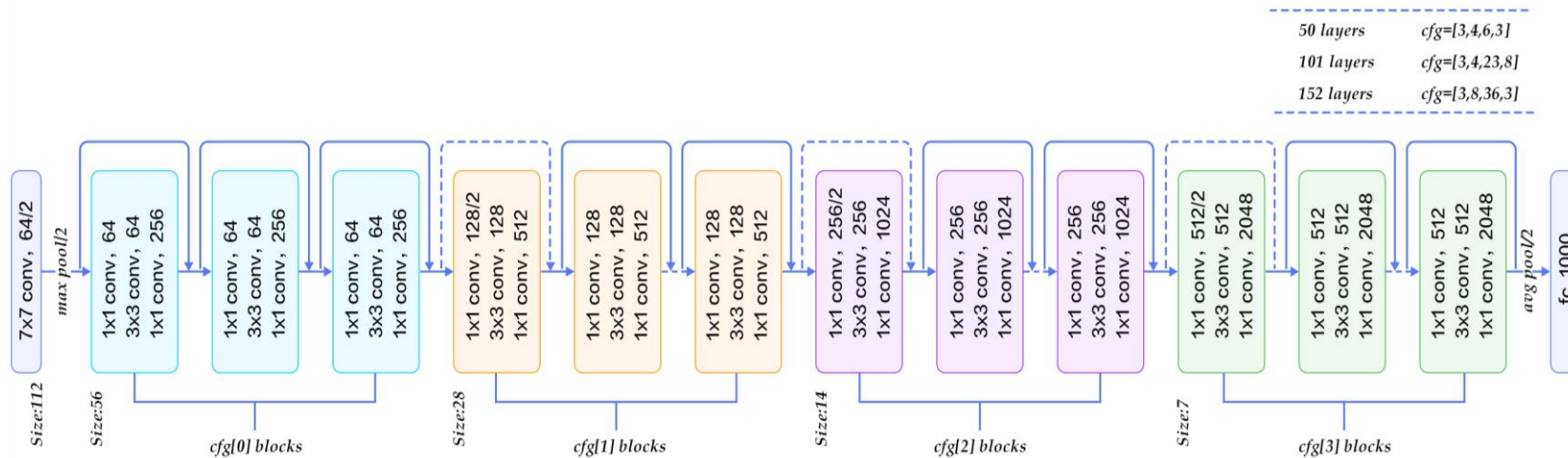
Contagem e identificação  
de leucócitos

## RESNET 50



A **ResNet** possui uma arquitetura que assenta em módulos de micro-arquitetura (*network-in-network architectures*).

Esta rede demonstrou que é possível treinar redes extremamente profundas utilizando o SGD padrão através do uso de módulos residuais.



Fonte: <https://www.codeproject.com/Articles/1248963/Deep-Learning-using-Python-plus-Keras-Chapter-Re>



Implementação

CNN

Redes Neuronais

Contagem e identificação  
de leucócitos



A análise dos resultados centra-se nas **accuracy** e **loss** de treino e validação de um conjunto de modelos, bem como nos resultados com o dataset de teste.

## MODELOS COM *TRANSFER LEARNING*



No caso do treino com *transfer learning*, foi excluído o topo das redes. Utilizaram-se os pesos do treino no ImageNet, e foi então criada uma pequena rede com 3 camadas: uma *dense* com a ativação *ReLU*, uma camada de *Dropout* e uma camada de saída com a função de *Softmax* para 4 categorias.

```
model = models.Sequential()  
model.add(layers.Dense(256, activation='relu', input_dim=7 * 7 * 512))  
model.add(layers.Dropout(0.5))  
model.add(layers.Dense(4, activation='softmax'))
```



Número de epochs: 100  
Batch size: 16  
Tempo médio de treino: 00:09:59  
Rede mais rápida: VGG16 (00:05:47)  
Rede mais lenta: Xception (00:16:57)



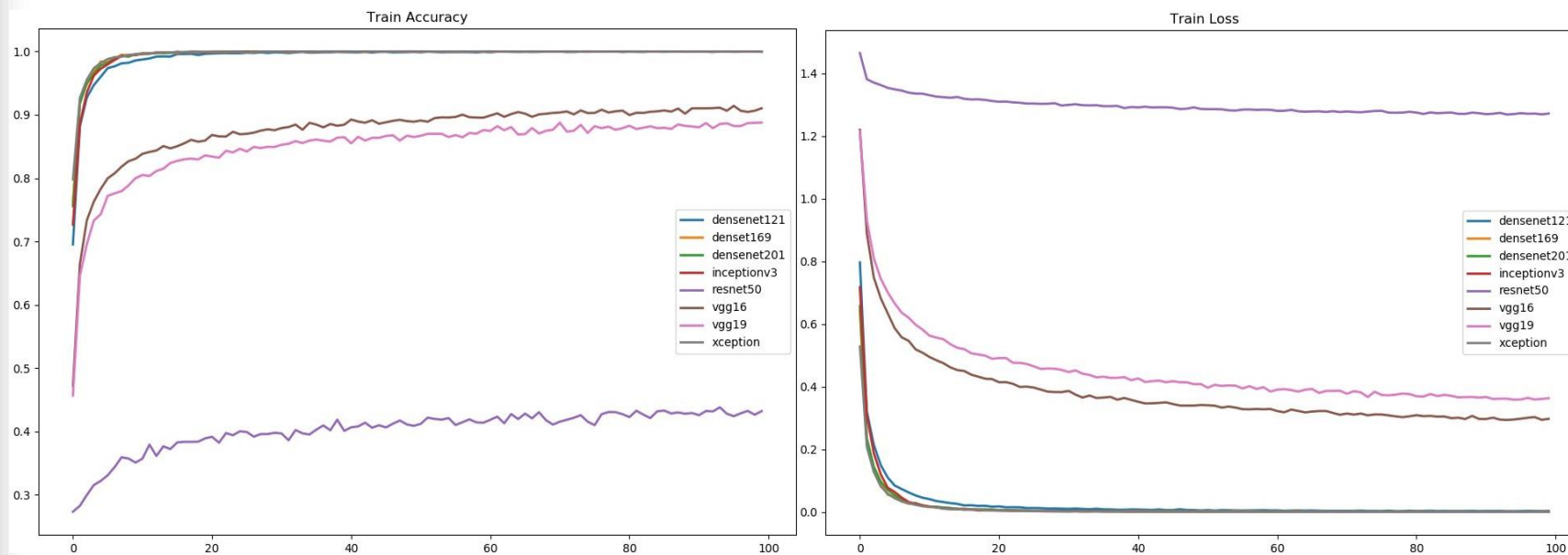
Implementação

CNN

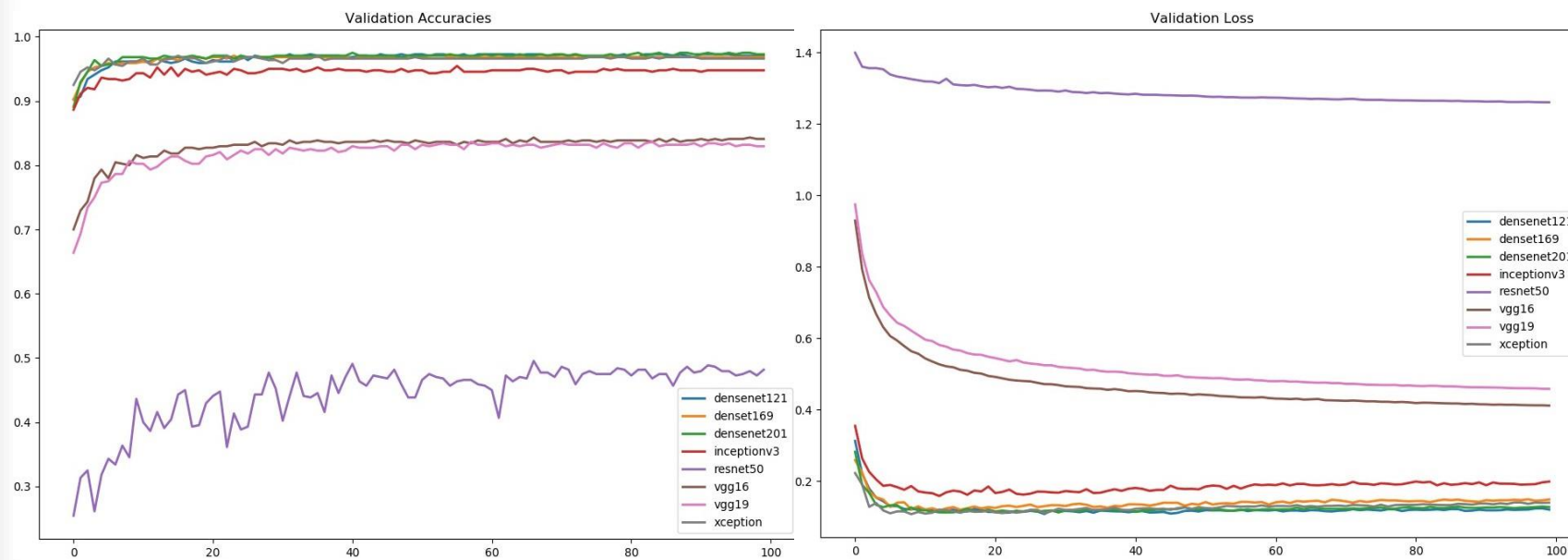
Redes Neurais

Contagem e identificação  
de leucócitos

# TRANSFER LEARNING: TRAIN ACCURACY AND TRAIN LOSS



# TRANSFER LEARNING: VALIDATION ACCURACY AND VALIDATION LOSS



Resultados

Implementação

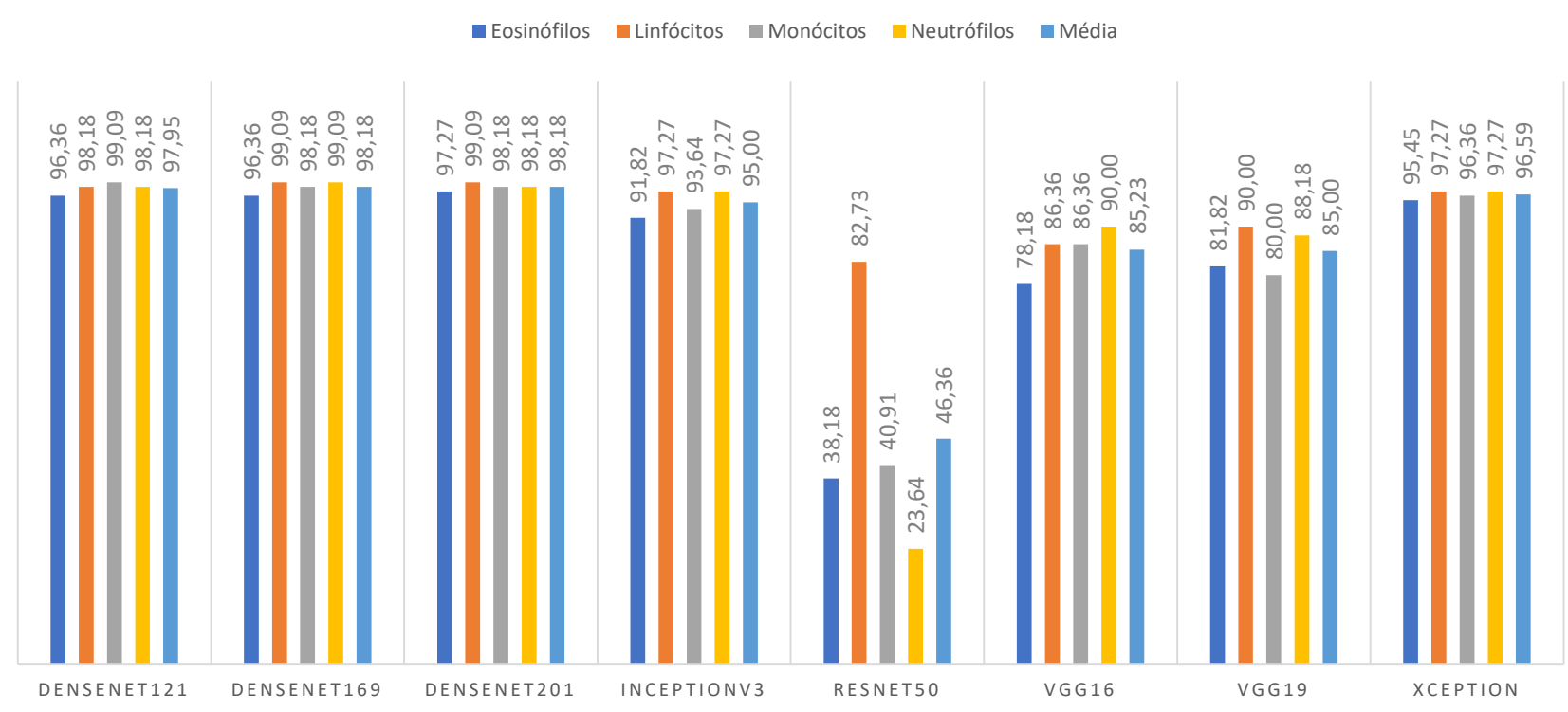
CNN

Redes Neurais



Contagem e identificação de leucócitos


# Conclusões

TRANSFER LEARNING: PERCENTAGENS DE ACERTO, NA CLASSIFICAÇÃO DAS IMAGENS DO DATASET DE TESTE, COM AS REDES TREINADAS POR 100 EPOCHS.



## MODELOS TREINADOS DE RAIZ

-  Nos modelos treinados de raiz é necessário treinar a totalidade da rede, sendo indispensável a especificação do número de classes ou categorias.
-  É um processo muito mais moroso do que o treino com *transfer learning*.

 Número de epochs: 100  
Batch Size: 16  
Tempo total de treino: >18:00:00  
Tempo médio de treino: 02:14:00  
Rede mais rápida: VGG16 (01:15:00)  
Rede mais lenta: Xception (03:24:00)



Resultados

Implementação

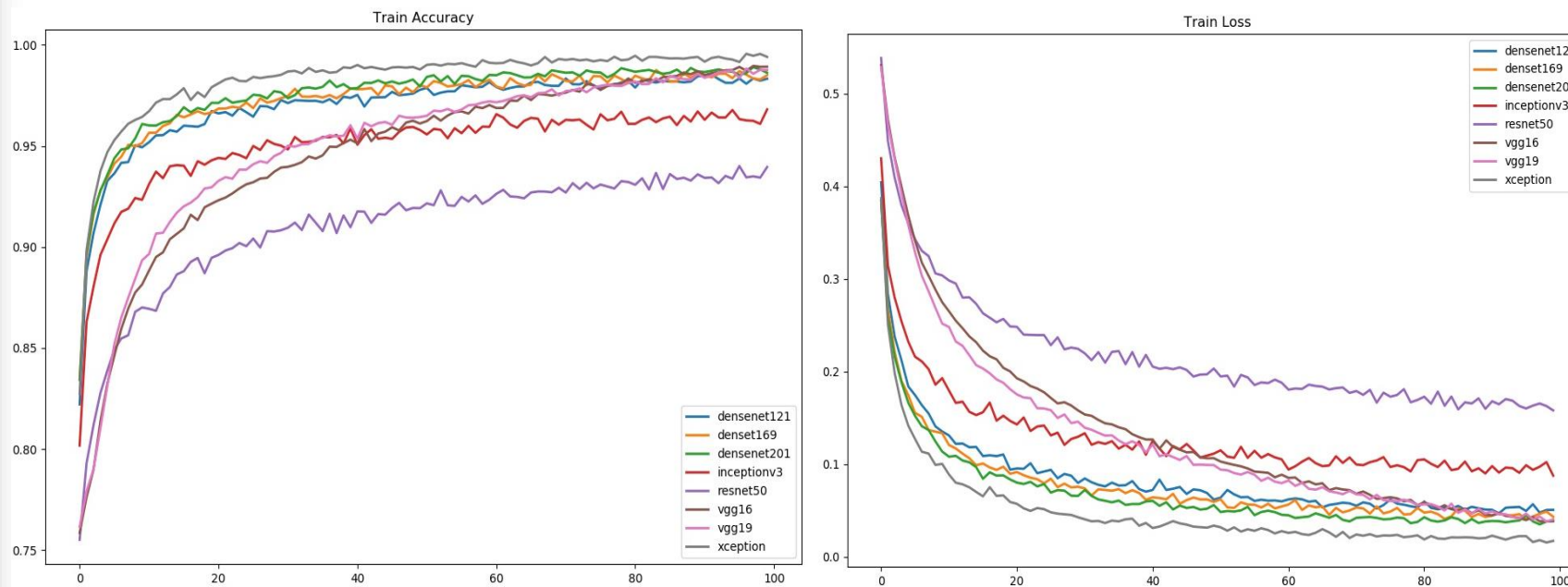
CNN

Redes Neurais

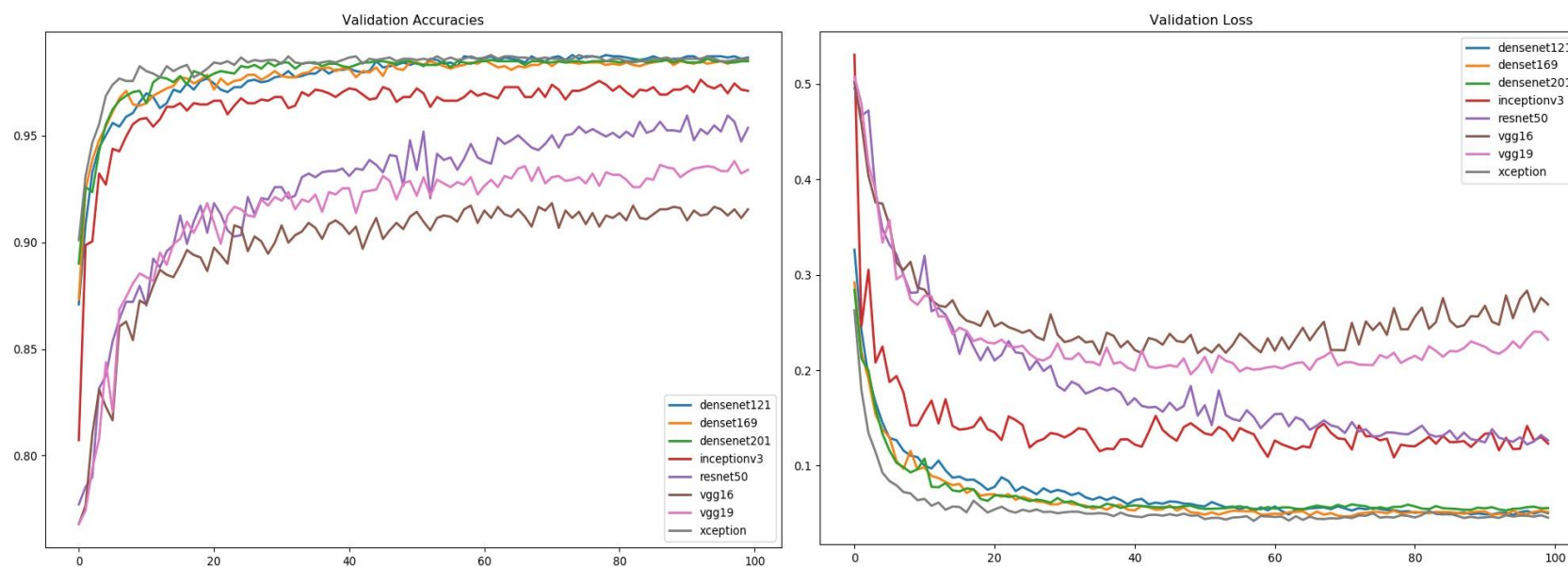
Contagem e identificação  
de leucócitos



# MODELOS TREINADOS DE RAIZ: TRAIN ACCURACY AND TRAIN LOSS



# MODELOS TREINADOS DE RAIZ: VALIDATION ACCURACY AND VALIDATION LOSS



Resultados

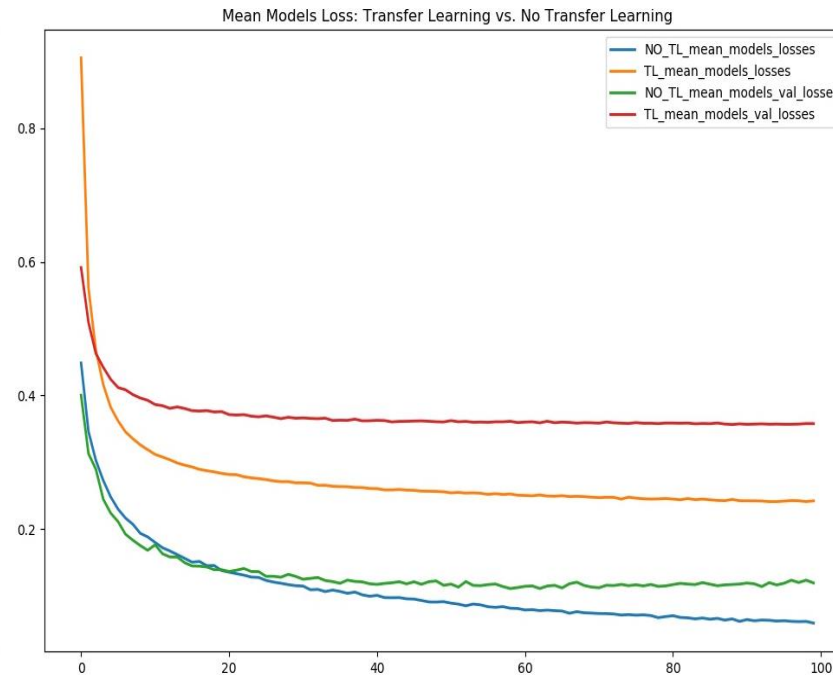
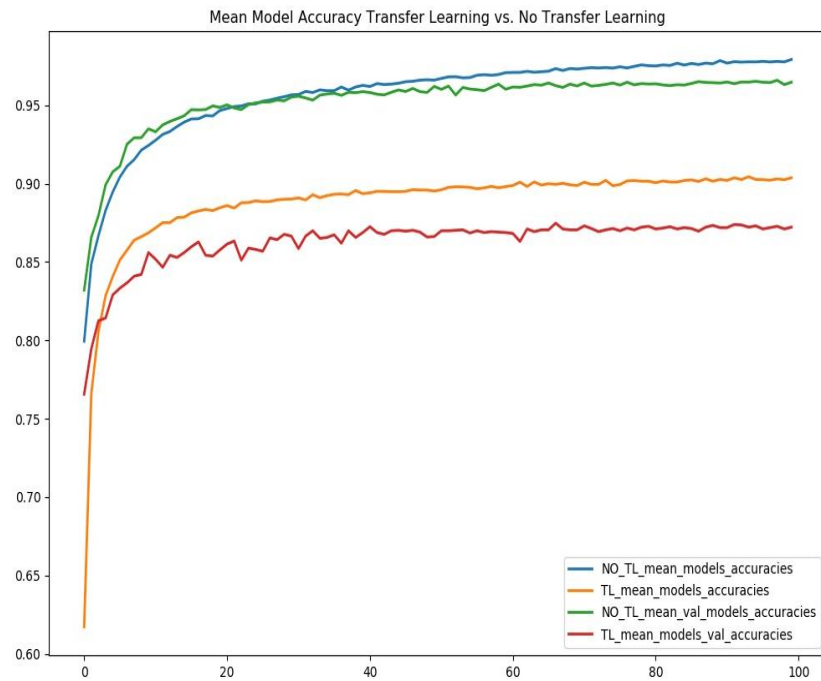
Implementação

CNN

Redes Neurais

Contagem e identificação  
de leucócitos

## MODELOS COM *TRANSFER LEARNING* VS. MODELOS TREINADOS DE RAIZ



Resultados

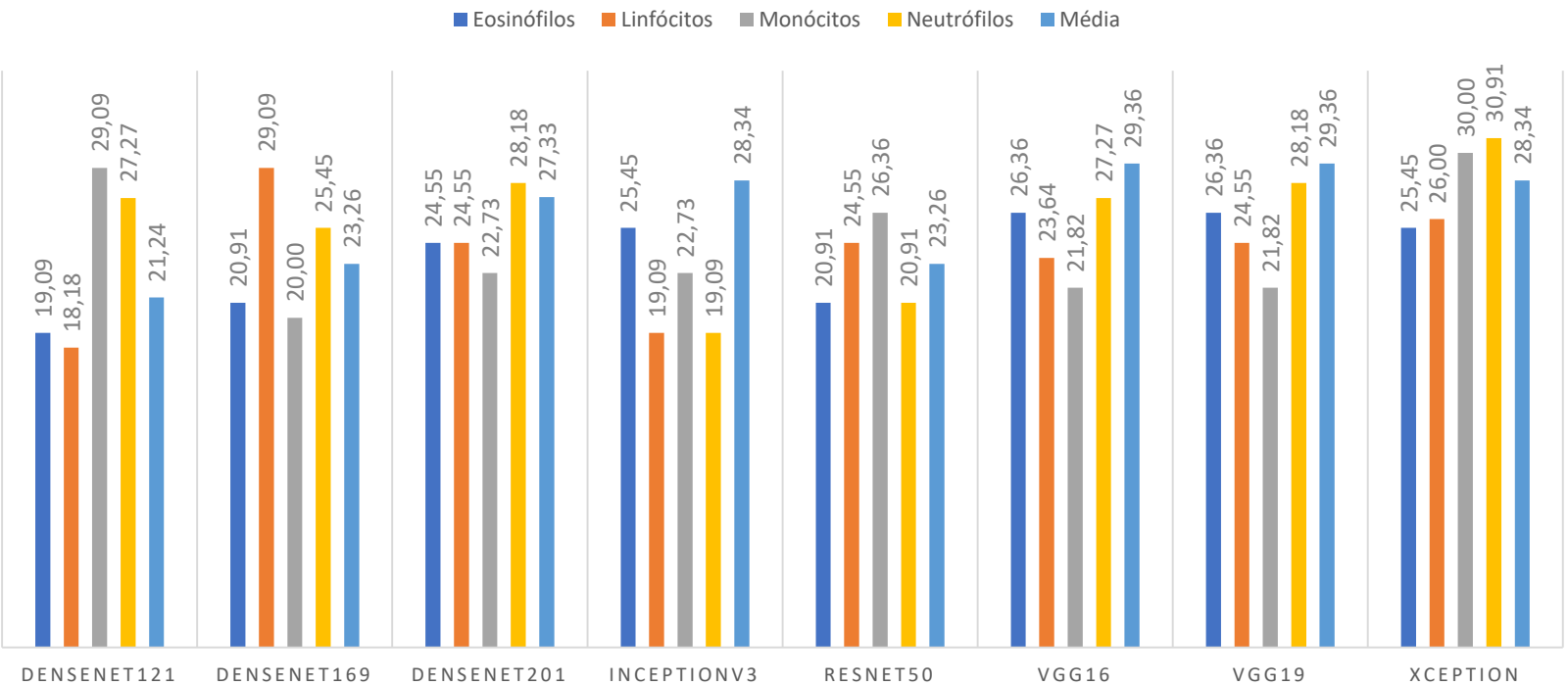
Implementação

CNN

Redes Neurais

Contagem e identificação  
de leucócitos

MODELOS TREINADOS DE RAIZ: PERCENTAGENS DE ACERTO, NA CLASSIFICAÇÃO DAS IMAGENS DO DATASET DE TESTE, COM AS REDES TREINADAS POR 100 EPOCHS.



 Resultados

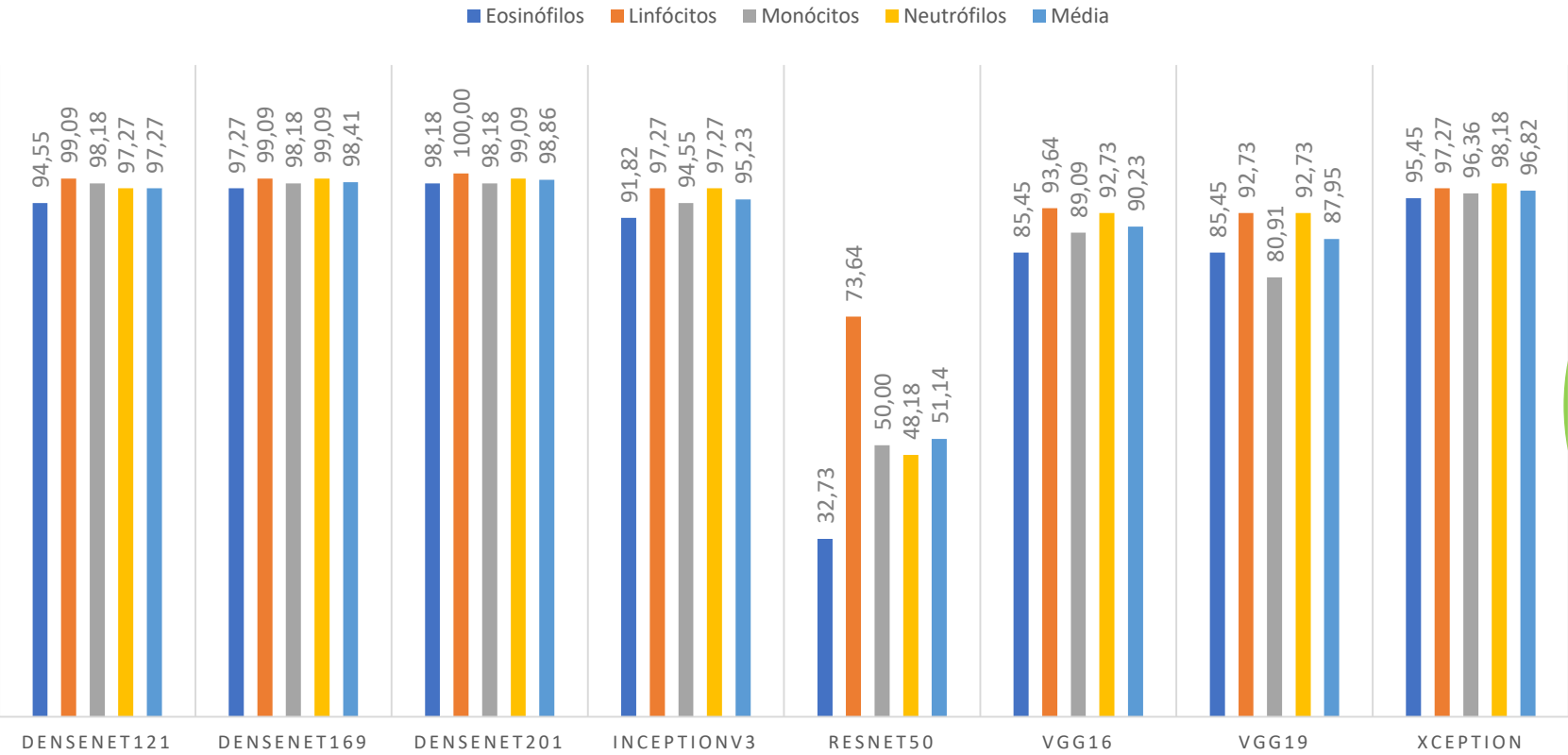
Implementação

CNN

Redes Neurais

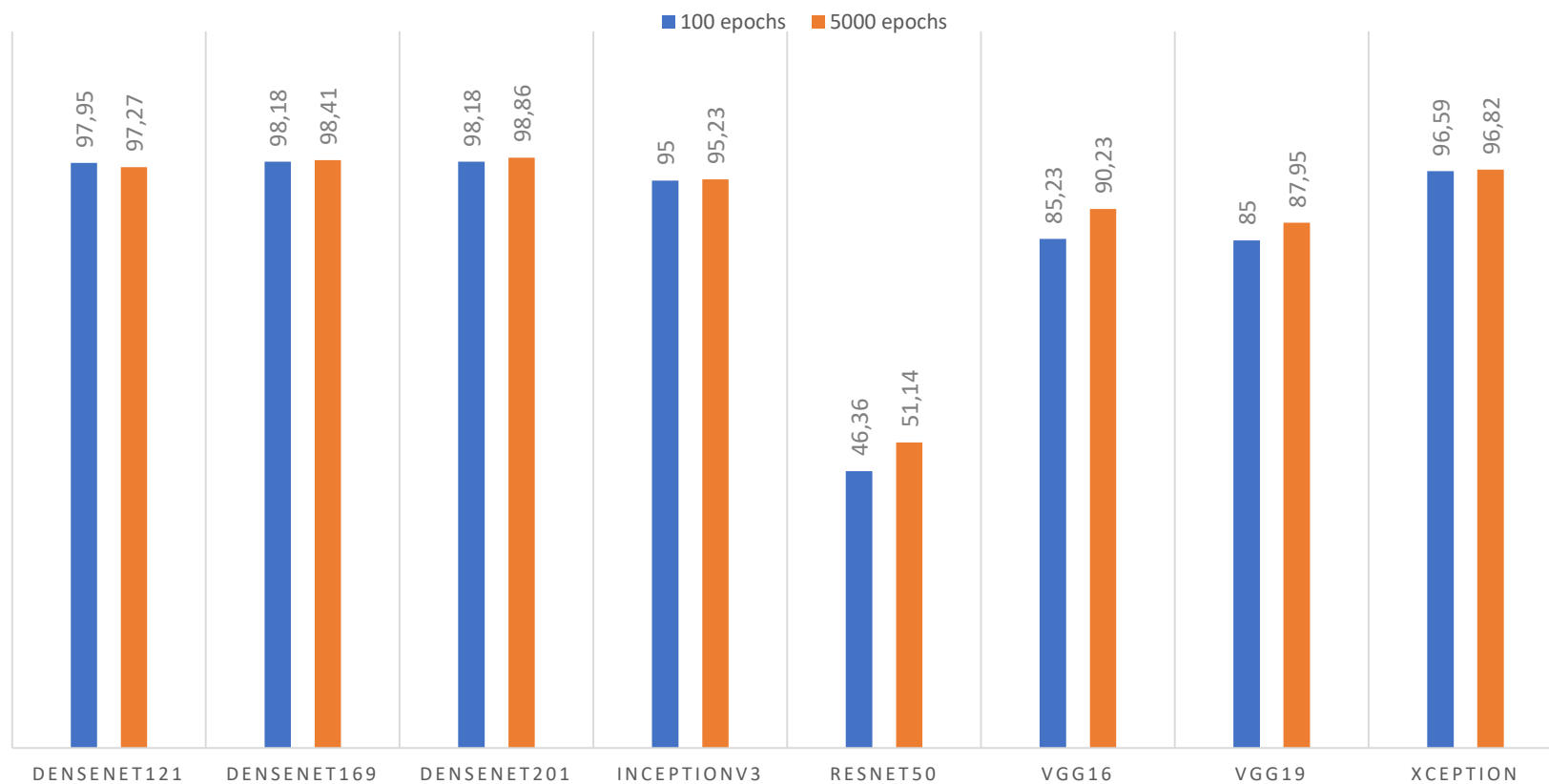
Contagem e identificação de leucócitos

TREINO DOS MODELOS COM *TRANSFER LEARNING* POR 5000 EPOCHS



## MODELOS COM TRANSFER LEARNING: 100 VS. 5000 EPOCHS

Percentagens de acerto em imagens do *dataset* de teste obtidas com modelos de *Transfer Learning* treinados por 100 e 5000 epochs





Apesar dos **elevados resultados de accuracy** e **baixa loss** dos modelos treinados de raiz, verificou-se que estes estavam a realizar *overfitting*.



As redes com **transfer learning** conseguiram, no seu conjunto, precisões de classificação de **87.8% contra 26.3%** das redes treinadas de raiz. As redes com melhores resultados foram as **DenseNet169** e **DenseNet201** ambas com **98.2% de acerto**. O tempo médio de treino dos modelos com **transfer learning** foi de **9 minutos e 59 segundos**, sendo necessárias, em média, **2 horas e 14 minutos para treinar um modelo de raiz**.



A solução para a melhoria dos resultados nos modelos treinados de raiz passaria pela utilização de **regularização** (L2 e L1, dropout e *early stopping*).



Os resultados dos modelos com **transfer learning** treinados por 5000 *epochs* mostram um **acréscimo 1.68%** na generalidade, na percentagem de acerto destas redes, quando testadas.





**Limitações:** o **tempo** necessário para testar os modelos e, numa fase inicial, alguma indisponibilidade de **recursos computacionais**, uma vez que o acesso a GPUs poderosos acelera vertiginosamente o processo de treino.



É concebível que com um conjunto de imagens significativamente maior e com o acoplamento de um mecanismo computacional de identificação e segmentação de imagens (podendo também este ser uma rede neuronal) se conseguisse **automatizar a identificação, classificação e contagem de todos os tipos de leucócitos** existentes a partir de imagens de microscopia.



Obrigado pela atenção

