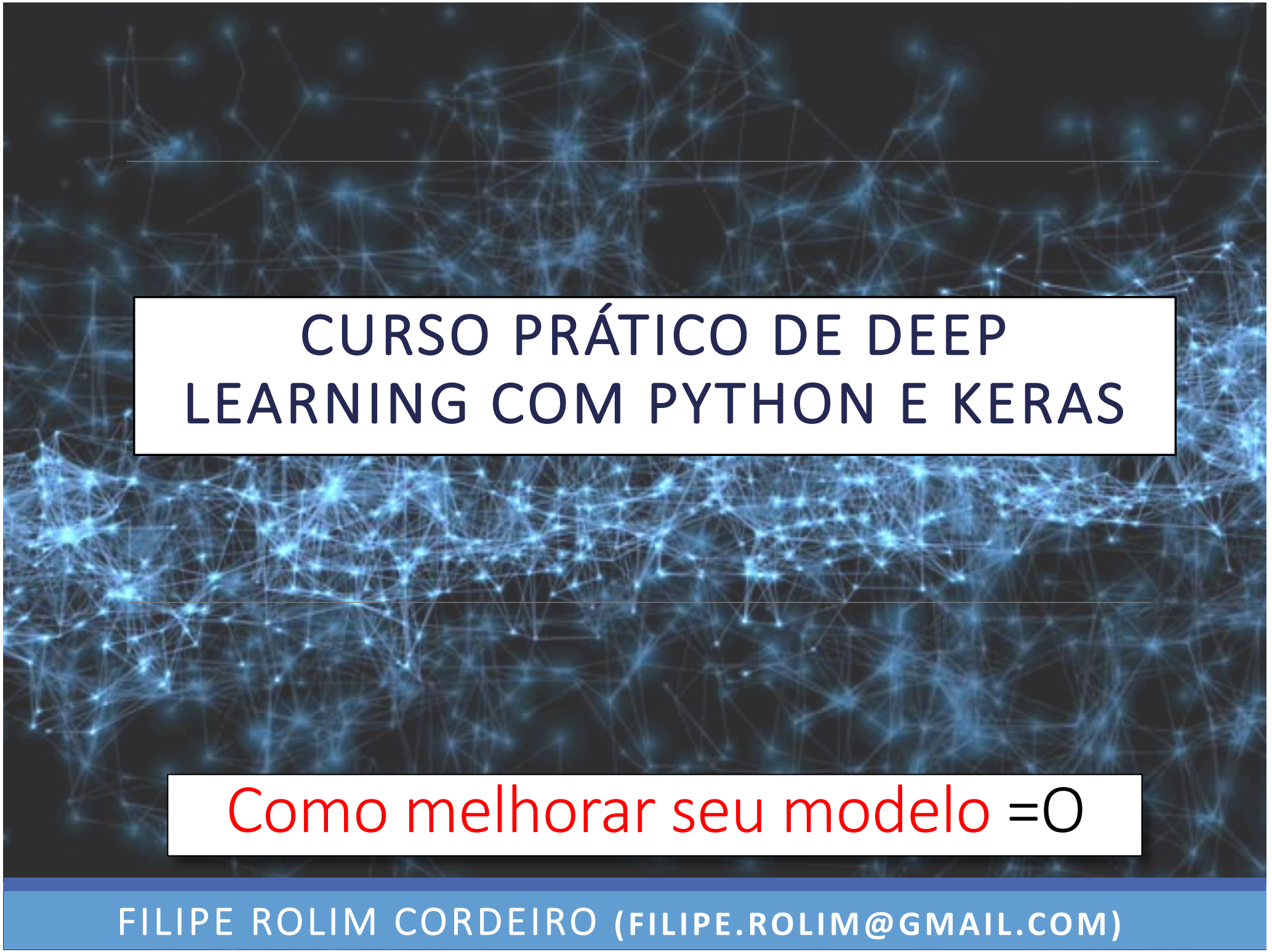


FILIPE ROLIM CORDEIRO (FILIPE.ROLIM@GMAIL.COM)

Conjuntos de treinamento/teste

- Divisões usadas: 70/30, 60/20/20, 50/25/25
- Big data (1.000.000)
  - 98/1/1/

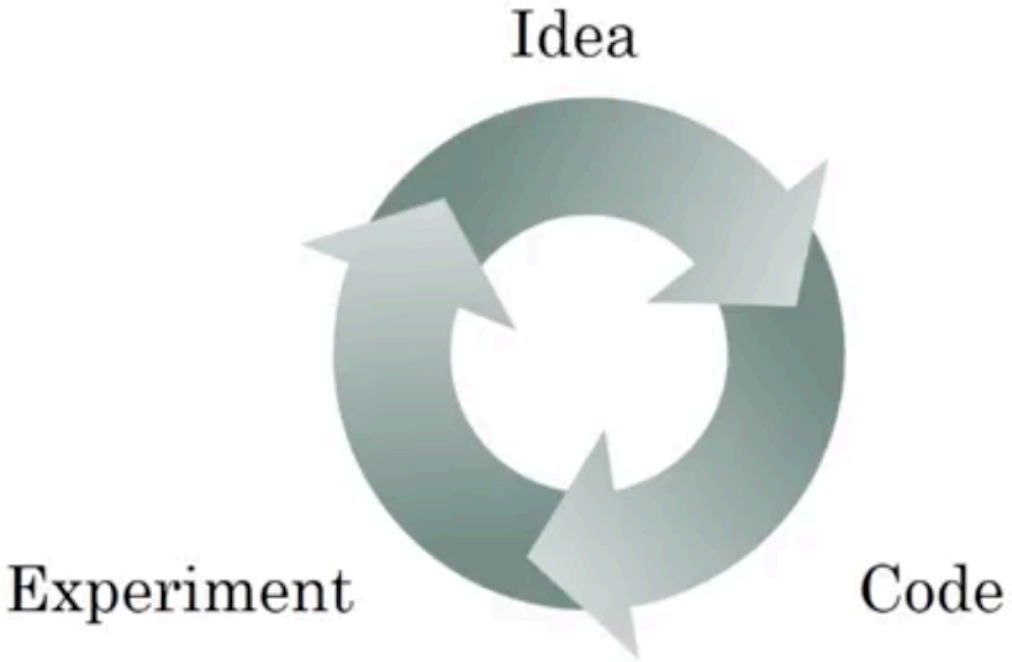
Distribuição incorreta

Conjunto de treino: figuras de gatos de webpages

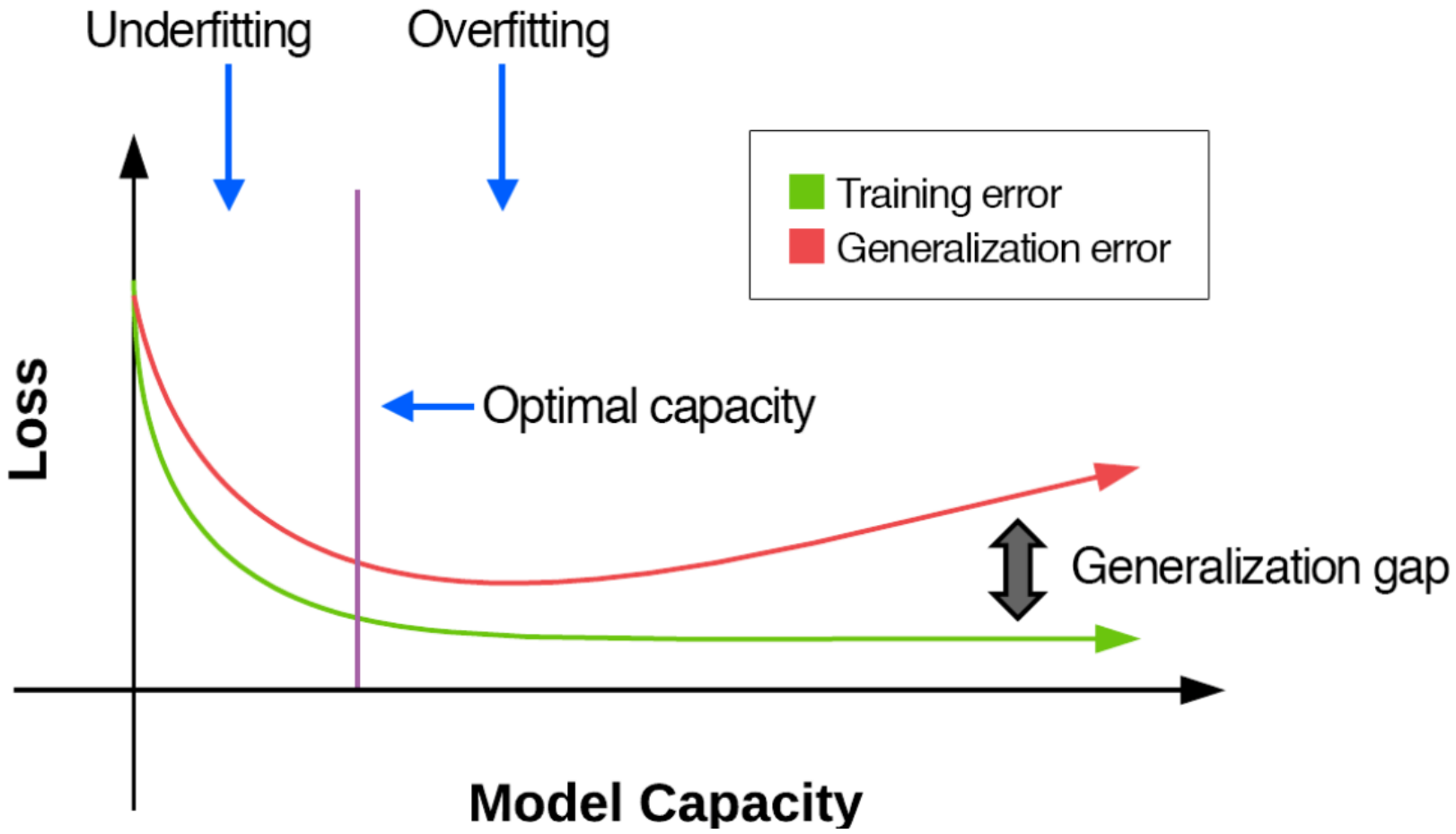
Conjunto de Valid./teste: Figuras de gatos usando seu aplicativo

ML é um processo iterativo

- # camadas
- # unidades escondidas
- Taxas de aprendizado
- Funções de ativação



Overfitting e Underfitting

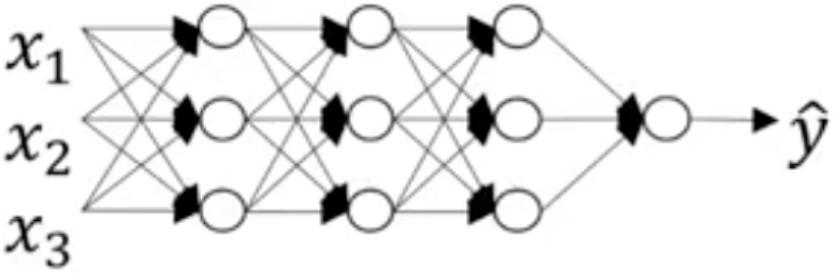


Regularização

$$L = \frac{1}{N}\sum_{i=1}^{N}[-log(e^{s_{y_i}} / \sum_j e^{s_j})] + \lambda \sum_i \sum_j W_{i,j}^2$$

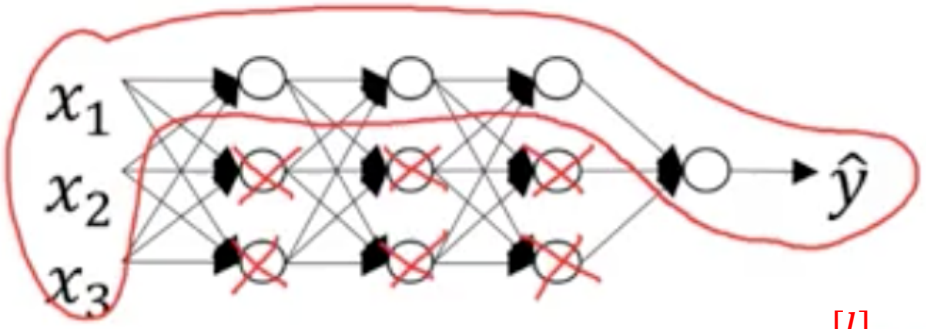
- Adiciona uma penalidade para valores altos de pesos
  - Parâmetro de ajuste λ

Por que a regularização previne overfitting?



$$L = \frac{1}{N}\sum_{i=1}^{N}[-log(e^{s_{y_i}}/\sum_j e^{s_j})] + \lambda \sum_i \sum_j W_{i,j}^2$$

Por que a regularização previne overfitting?



$$L = \frac{1}{N}\sum_{i=1}^{N}[-log(e^{s_{y_i}}/\sum_j e^{s_j})] + \lambda \sum_i \sum_j W_{i,j}^2$$

$$w^{[l]} \approx 0$$

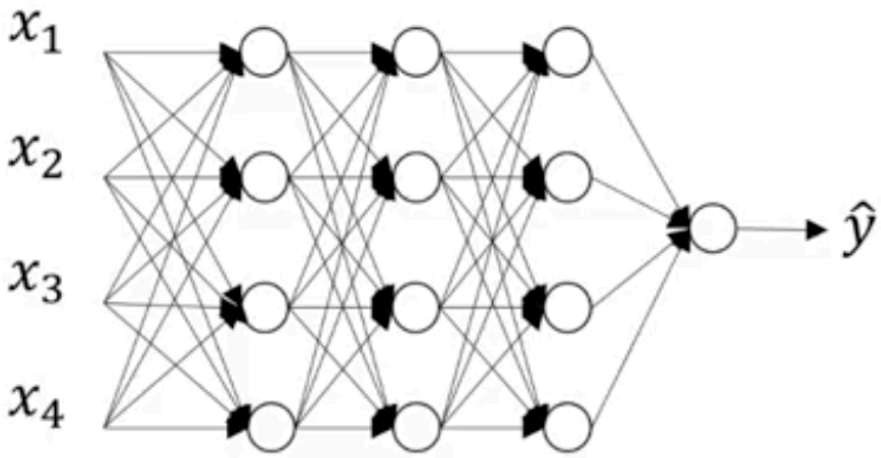
Regularização no Keras

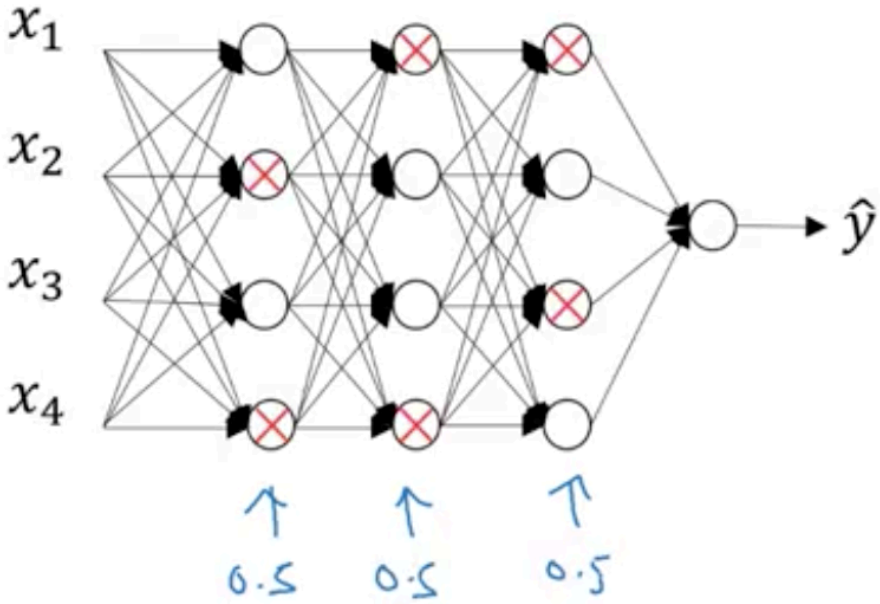
```
from keras import regularizers
model.add(Dense(64, input_dim=64,
                kernel_regularizer=regularizers.l2(0.01),
                activity_regularizer=regularizers.l1(0.01)))
```

Exercício

- Adicionar regularização
  - LeNet_optimizer.ipynb

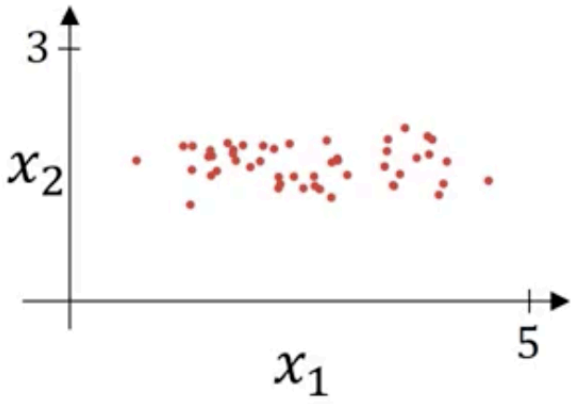
Dropout

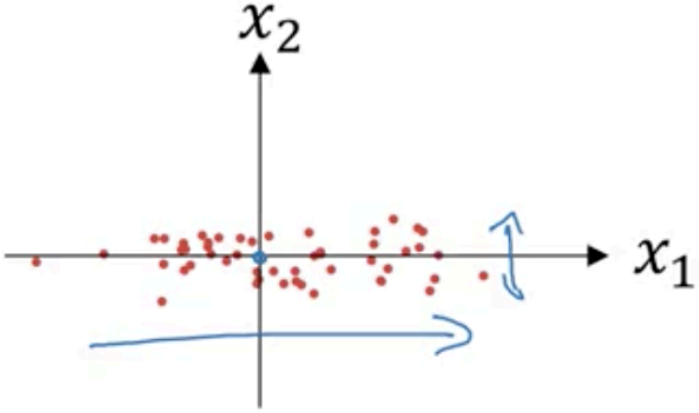


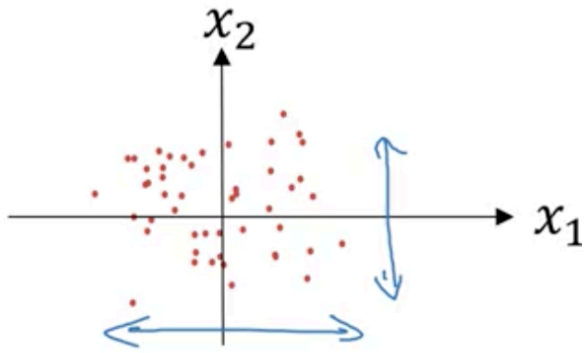


Normalizando Conjunto de Treino

- Média zero







Subtract mean:

$$\mu = \frac{1}{m} \sum_{i=1}^{m} x^{(i)}$$

$$x := x - \mu$$

Normalize variance

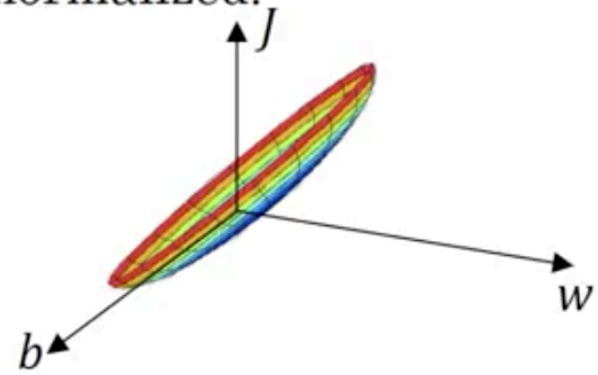
$$\sigma^2 = \frac{1}{m} \sum_{i=1}^{m} x^{(i)} **2$$

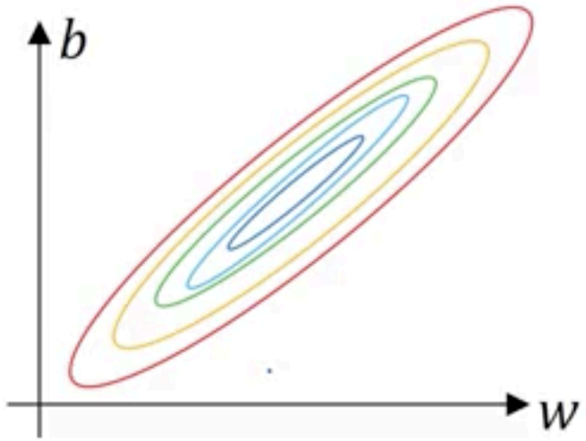
↖ element-wise

$$x /= \sigma^2$$

Por que normalizar entradas?

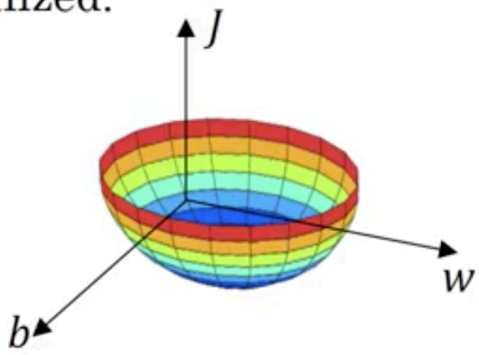
Unnormalized:

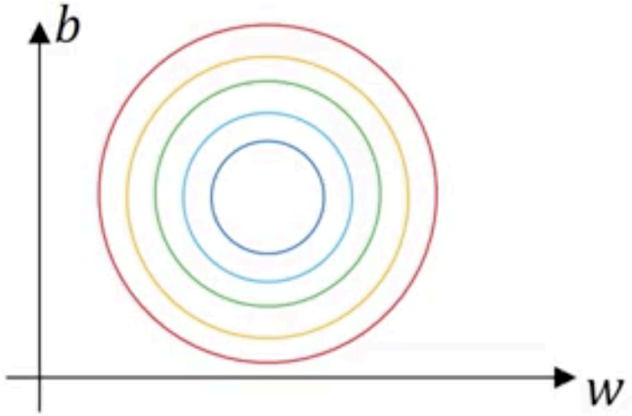




$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}\left(\hat{y}^{(i)}, y^{(i)}\right)$$

Normalized:



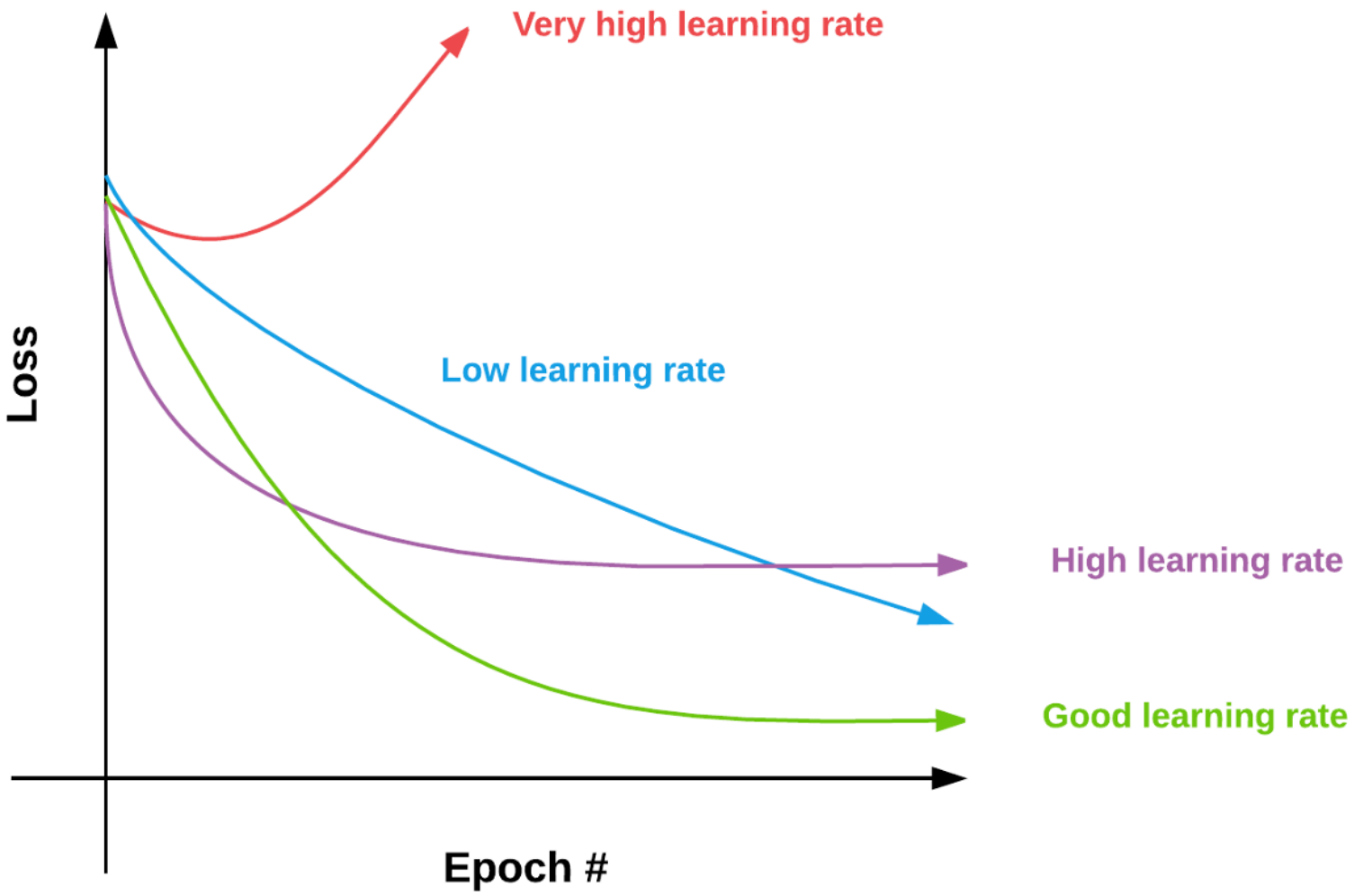


Métodos de Otimização

- SGD
- Adagrad
- Adadelta
- RMSProp
- Adam
- NAdam

Taxa de Aprendizado

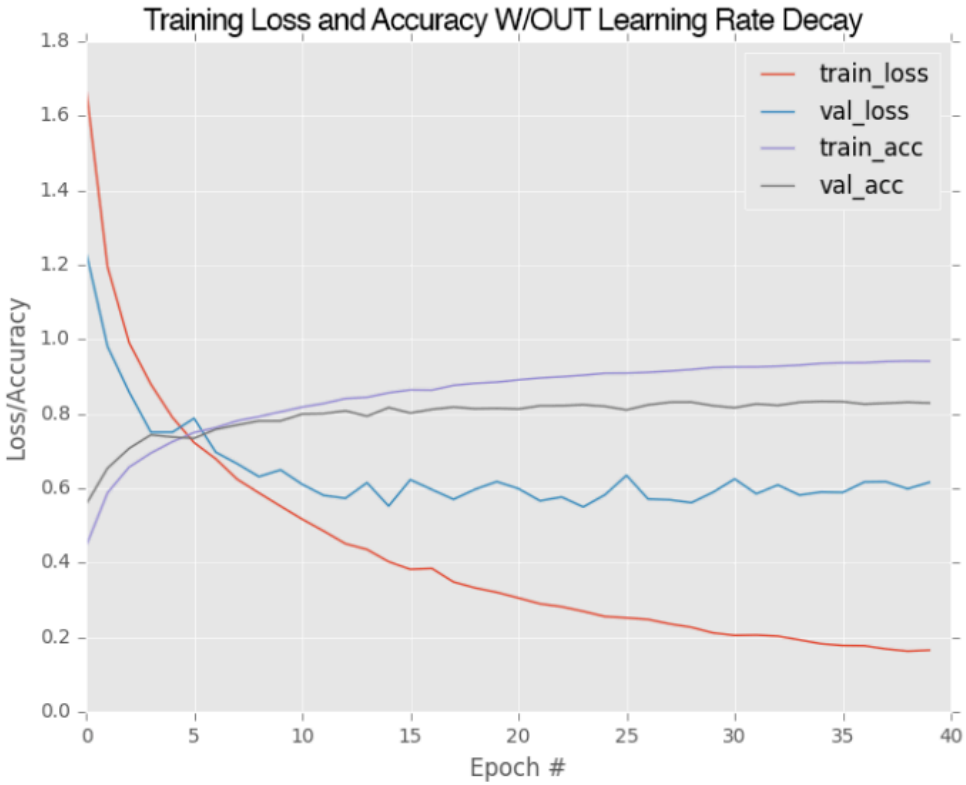
Efeitos da taxa de aprendizado



Learning Rate Decay

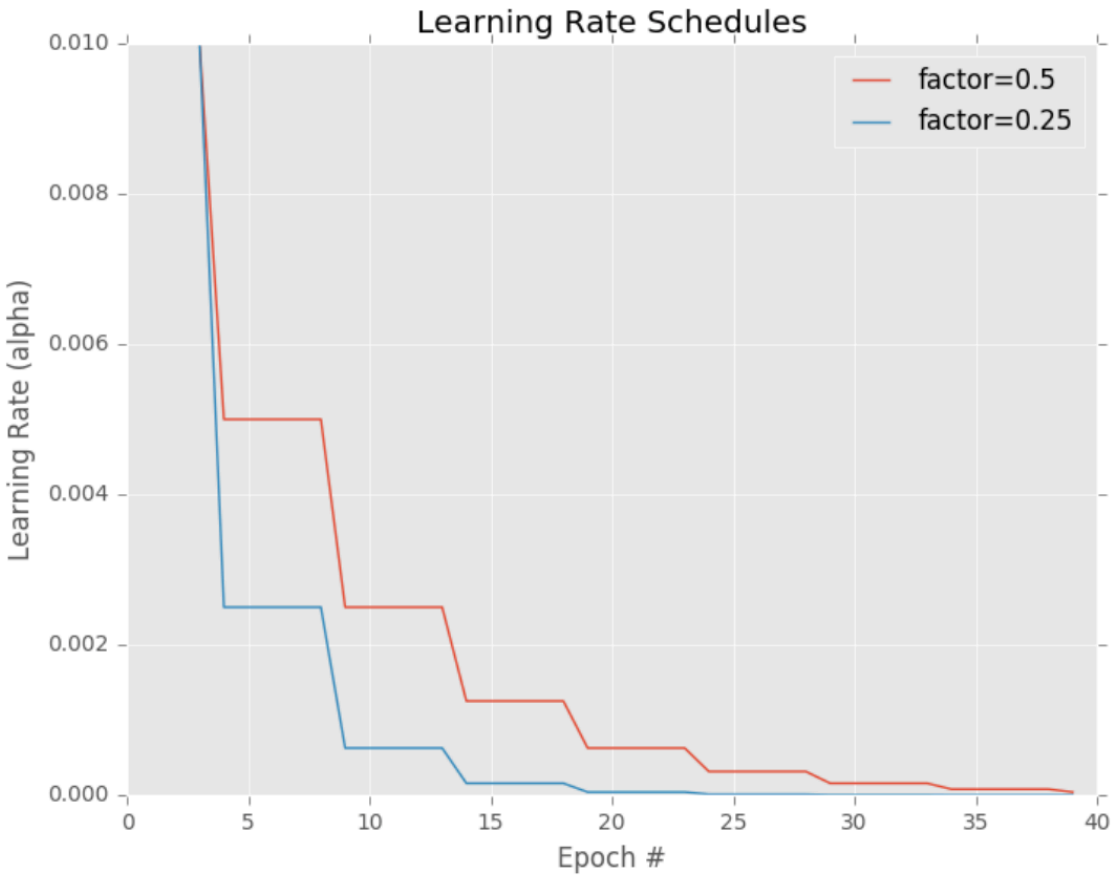
```
# initialize the optimizer and model
print("[INFO] compiling model...")
opt = SGD(lr=0.01, decay=0.01 / 40, momentum=0.9, nesterov=True)
model = MiniVGGNet.build(width=32, height=32, depth=3, classes=10)
model.compile(loss="categorical_crossentropy", optimizer=opt,
     metrics=["accuracy"])
```

Epoch	Learning Rate (α)
0	0.01000
1	0.00836
2	0.00719
...	...
37	0.00121
38	0.00119
39	0.00116


Training Loss and Accuracy W/OUT Learning Rate Decay
train_loss
val_loss
train_acc
val_acc
Loss/Accuracy
Epoch #
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
1.8
0
5
10
15
20
25
30
35
40


Training Loss and Accuracy WITH Learning Rate Decay
train_loss
val_loss
train_acc
val_acc
Loss/Accuracy
Epoch #
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
1.8
0
5
10
15
20
25
30
35
40

Step-based Decay



Exercício

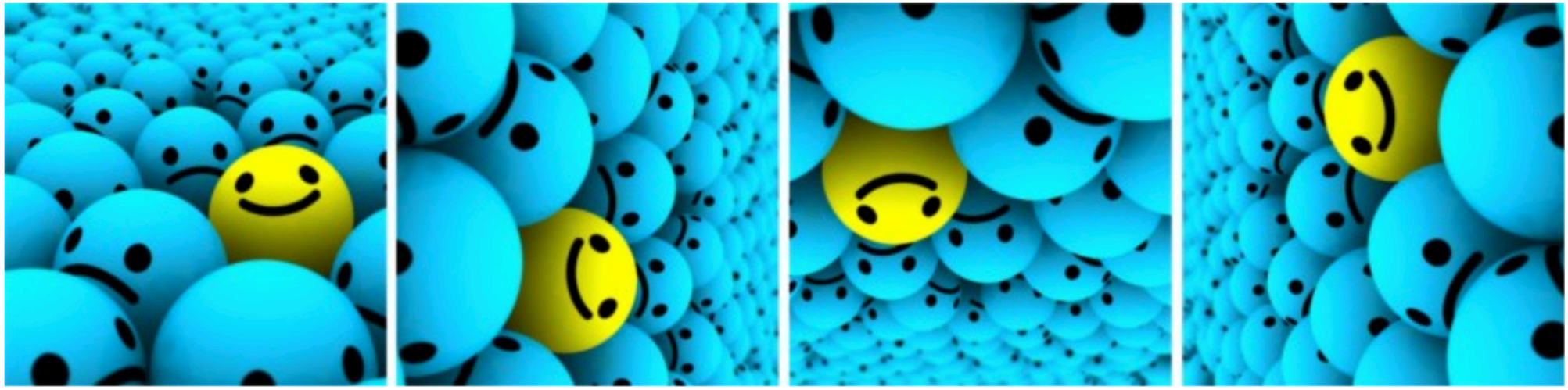
- Adicionar no LeNet
  - Dropout
  - Batch normalization
  - Usar otimizador diferente de SGD
  - Usar learning rate com decaimento

Data Augmentation

- Aplicar transformações na imagem:
  - Translação
  - Rotação
  - Escala
  - Flips
  - ...

Flip

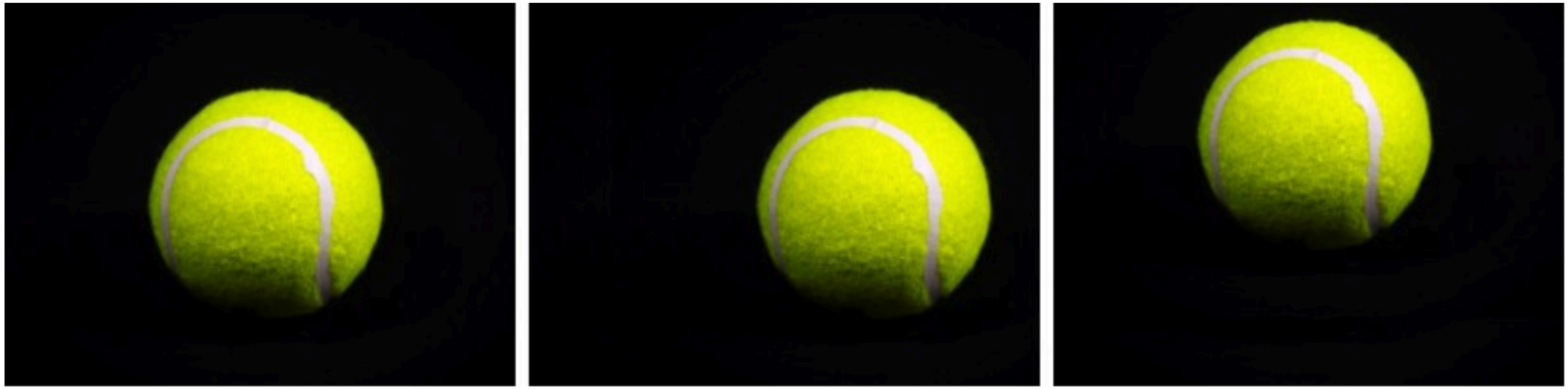
Rotação



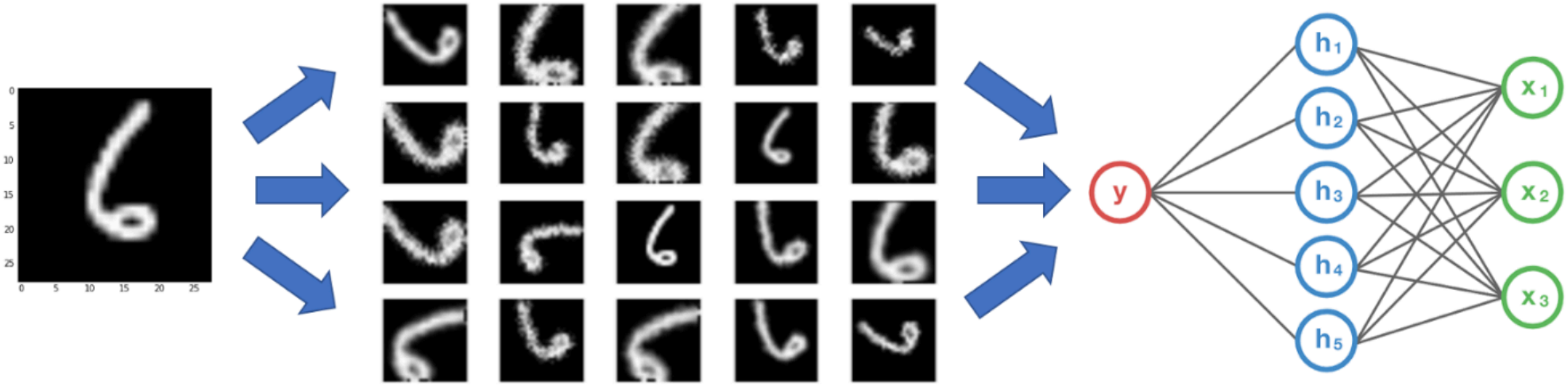
Escala

Crop

Translação



Ruído Gaussiano


0
5
10
15
20
25
0 5 10 15 20 25
y
h1
h2
h3
h4
h5
x1
x2
x3

Data Augmentation no Keras

```
from keras.preprocessing.image import ImageDataGenerator

generator = ImageDataGenerator(rescale = 1./255,
                               width_shift_range=0.1,
                               height_shift_range=0.1,
                               rotation_range = 20,
                               shear_range = 0.3,
                               zoom_range = 0.3,
                               horizontal_flip = True)
```

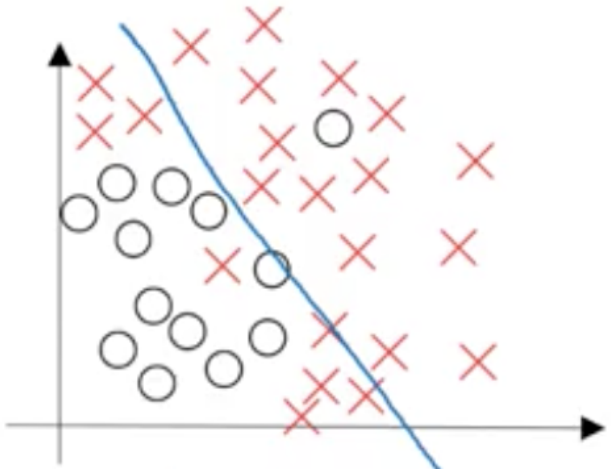
```
train = generator.flow_from_directory('../data/generator',
                                      target_size = (128, 128),
                                      batch_size = 32,
                                      class_mode = 'binary')
```

```
H = model.fit_generator(
  aug.flow(trainX, trainY, batch_size=BS),
  validation_data=(testX, testY),
  steps_per_epoch=len(trainX) // BS,
  epochs=EPOCHS, verbose=1)
```

Data augmentation

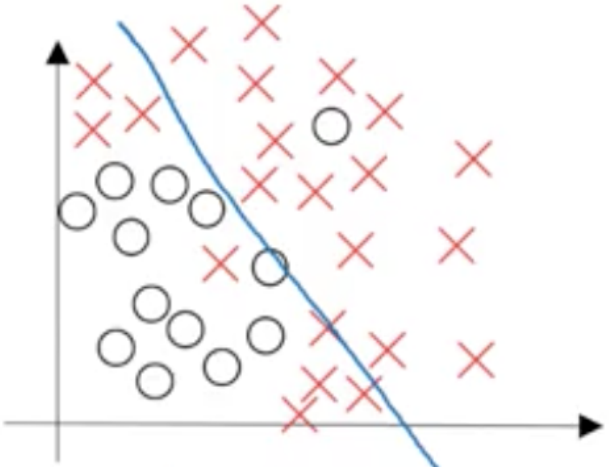
- Código
  - Augmentation.ipynb

Bias x Variância

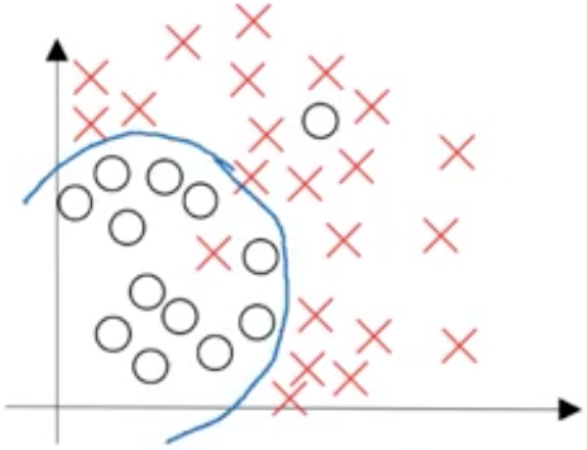


Bias alto

Bias x Variância

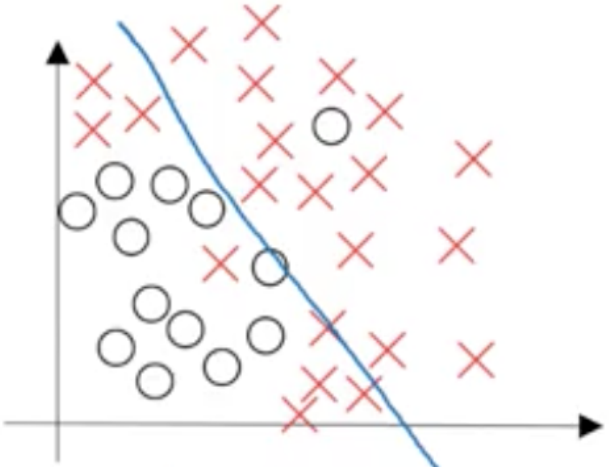


Bias alto

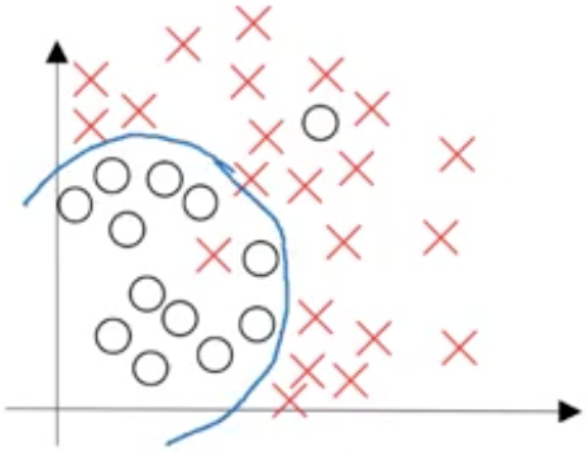


”Ok”

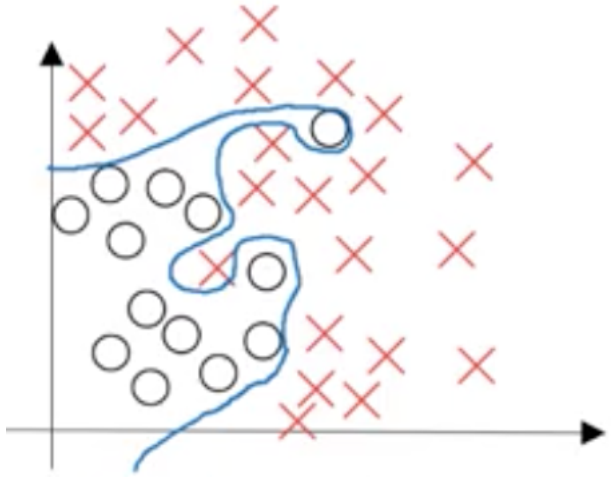
Bias x Variância



Bias alto

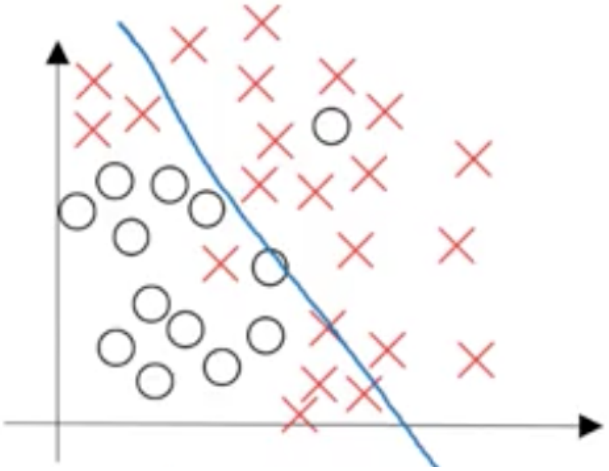


”Ok”



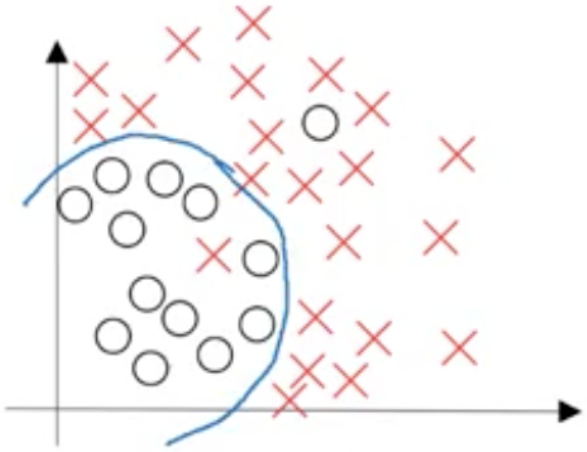
Variância alta

Bias x Variância

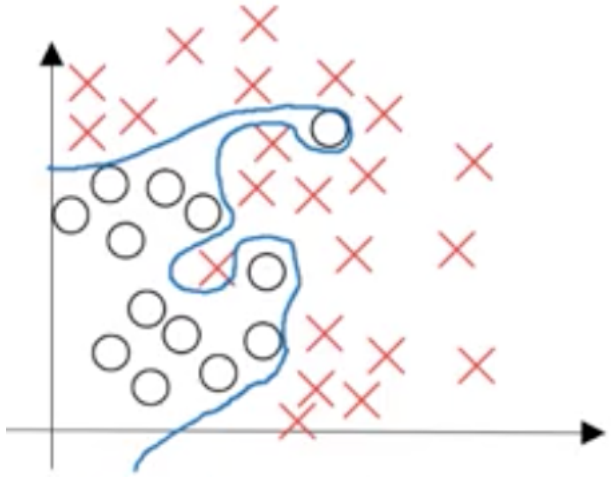


Bias alto

underfitting



”Ok”



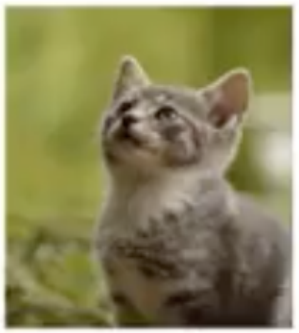
Variância alta

overfitting

Bias e Variância

Classificação de gato

y=1



y=0

Erro do conjunto de treino: 1%

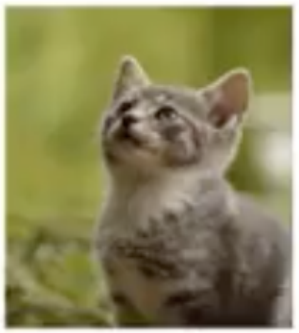
Erro do conjunto de teste: 11%

Humano: $\approx 0\%$

Bias e Variância

Classificação de gato

y=1



y=0

Erro do conjunto de treino: 1%

Erro do conjunto de teste: 11%

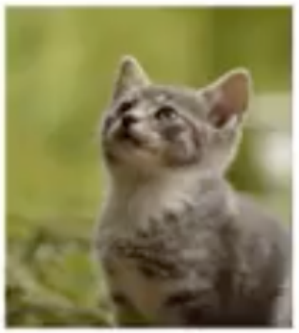
Variância alta

Humano: $\approx 0\%$

Bias e Variância

Classificação de gato

y=1



y=0

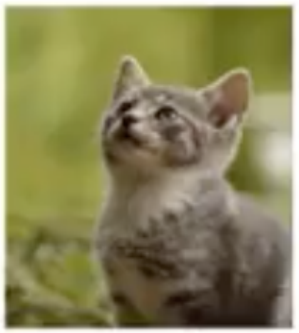
Erro do conjunto de treino:	1%	15%
Erro do conjunto de teste:	11%	16%
	Variância alta	

Humano: $\approx 0\%$

Bias e Variância

Classificação de gato

y=1



y=0

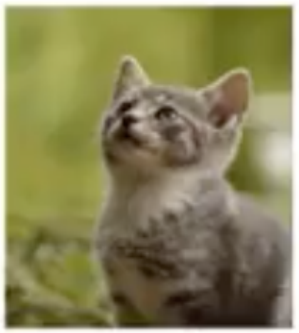
Erro do conjunto de treino:	1%	15%
Erro do conjunto de teste:	11%	16%
	Variância alta	Bias alto

Humano: $\approx 0\%$

Bias e Variância

Classificação de gato

y=1



y=0

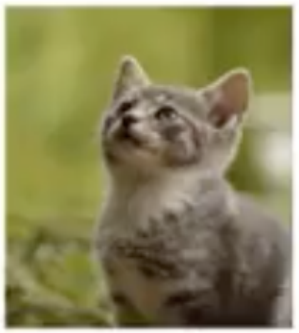
Erro do conjunto de treino:	1%	15%	15%
Erro do conjunto de teste:	11%	16%	30%
	Variância alta	Bias alto	

Humano: $\approx 0\%$

Bias e Variância

Classificação de gato

y=1



y=0

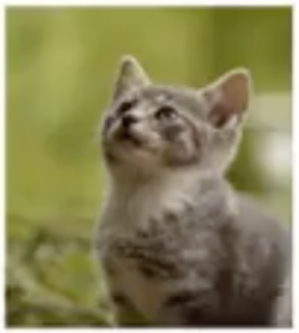
Erro do conjunto de treino:	1%	15%	15%
Erro do conjunto de teste:	11%	16%	30%
	Variância alta	Bias alto	- Variância alta - Bias alto

Humano: $\approx 0\%$

Bias e Variância

Classificação de gato

y=1



y=0

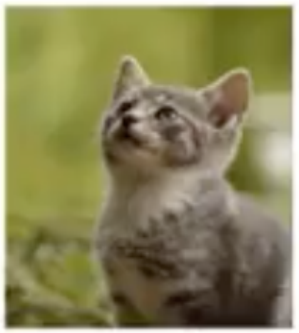
Erro do conjunto de treino:	1%	15%	15%	0.5%
Erro do conjunto de teste:	11%	16%	30%	1%
	Variância alta	Bias alto	- Variância alta - Bias alto	

Humano: $\approx 0\%$

Bias e Variância

Classificação de gato

y=1



y=0

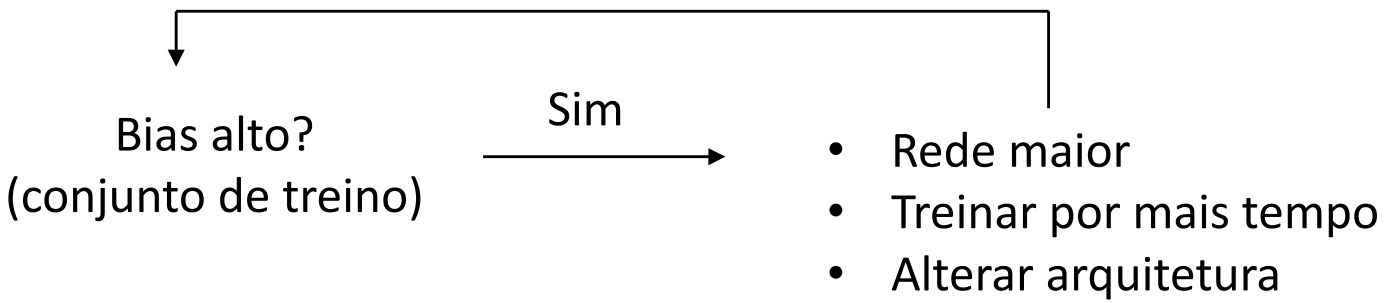
Erro do conjunto de treino:	1%	15%	15%	0.5%
Erro do conjunto de teste:	11%	16%	30%	1%
	Variância alta	Bias alto	- Variância alta - Bias alto	- Variância baixa - Bias baixo

Humano: $\approx 0\%$

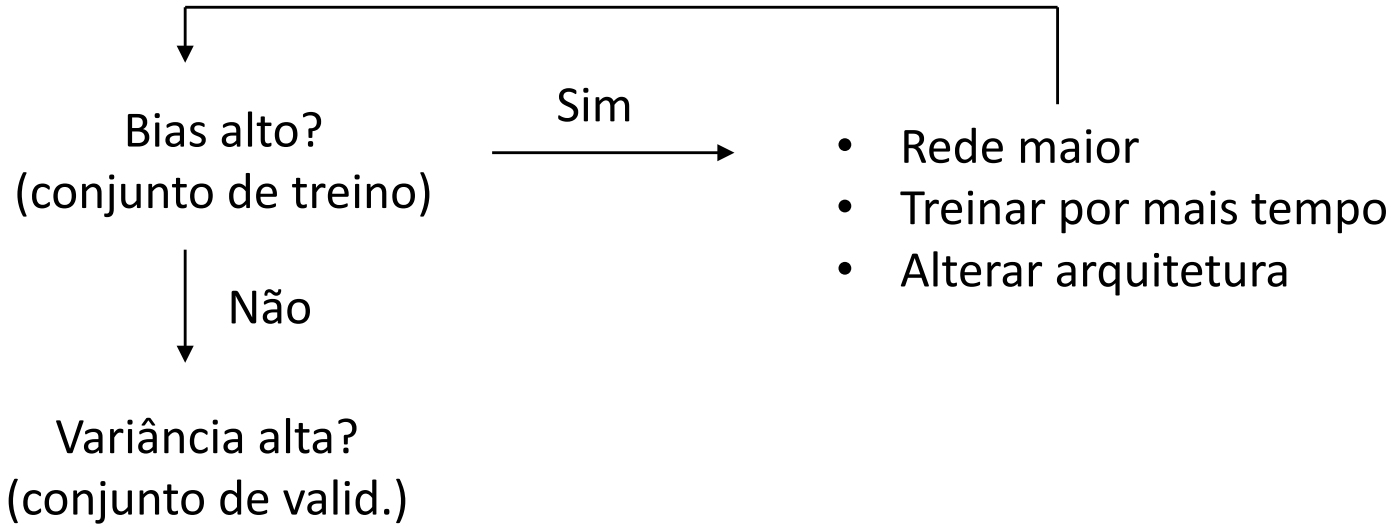
Receita básica para ajustar modelo

Bias alto?
(conjunto de treino)

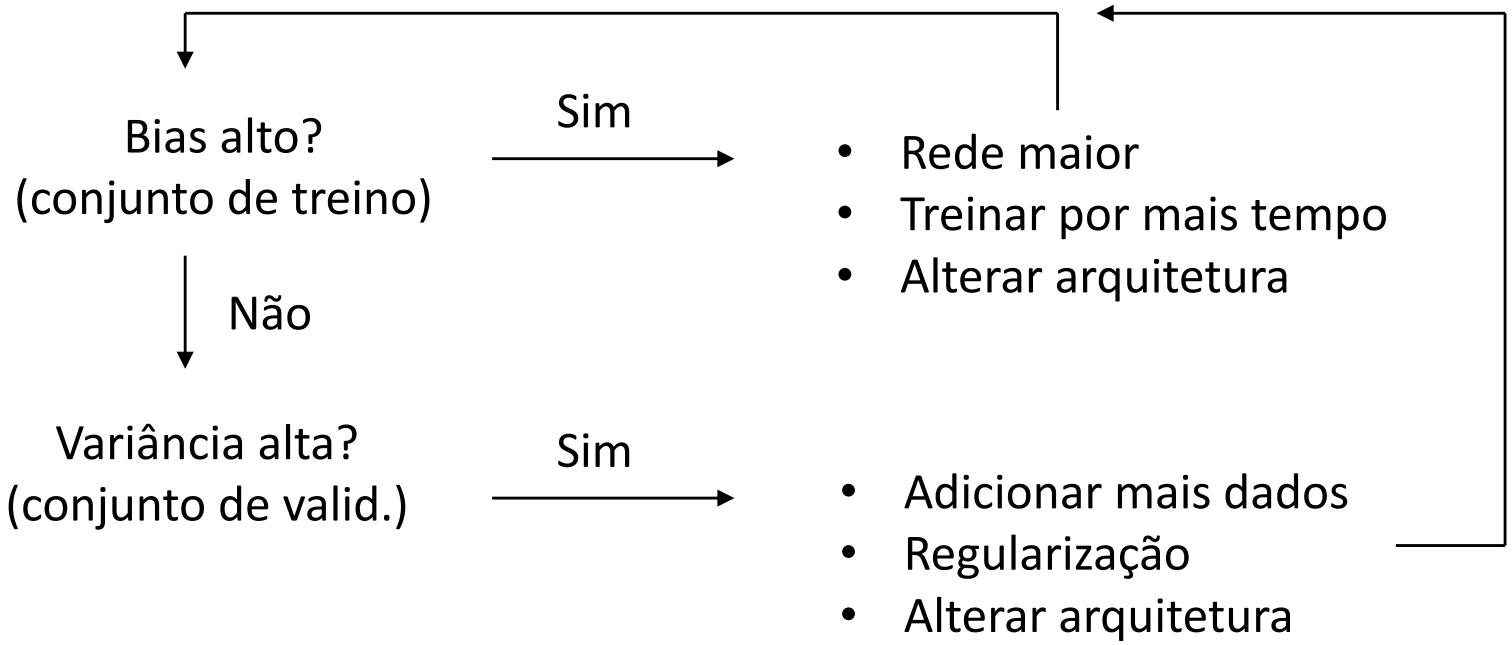
Receita básica para ajustar modelo



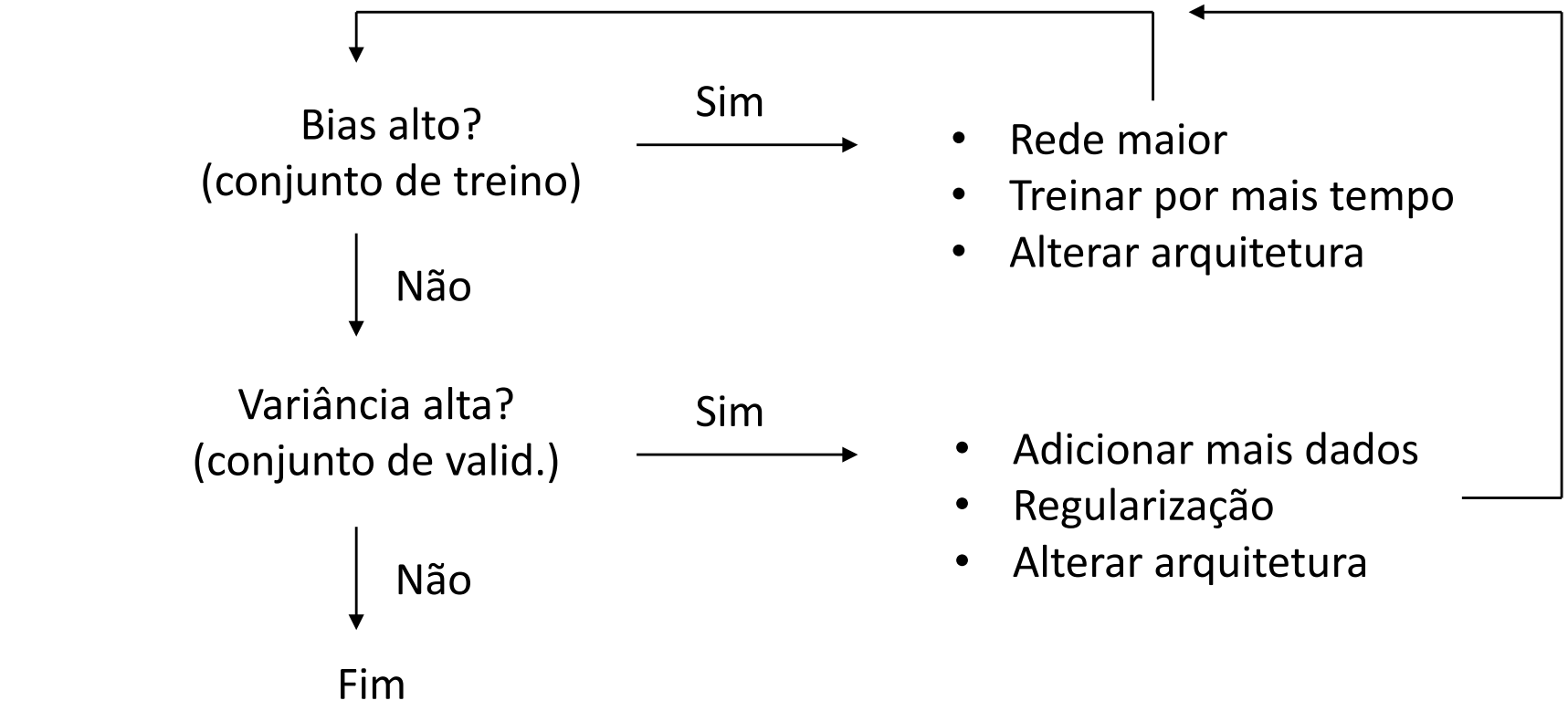
Receita básica para ajustar modelo



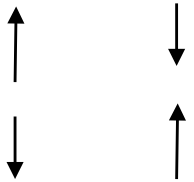
Receita básica para ajustar modelo



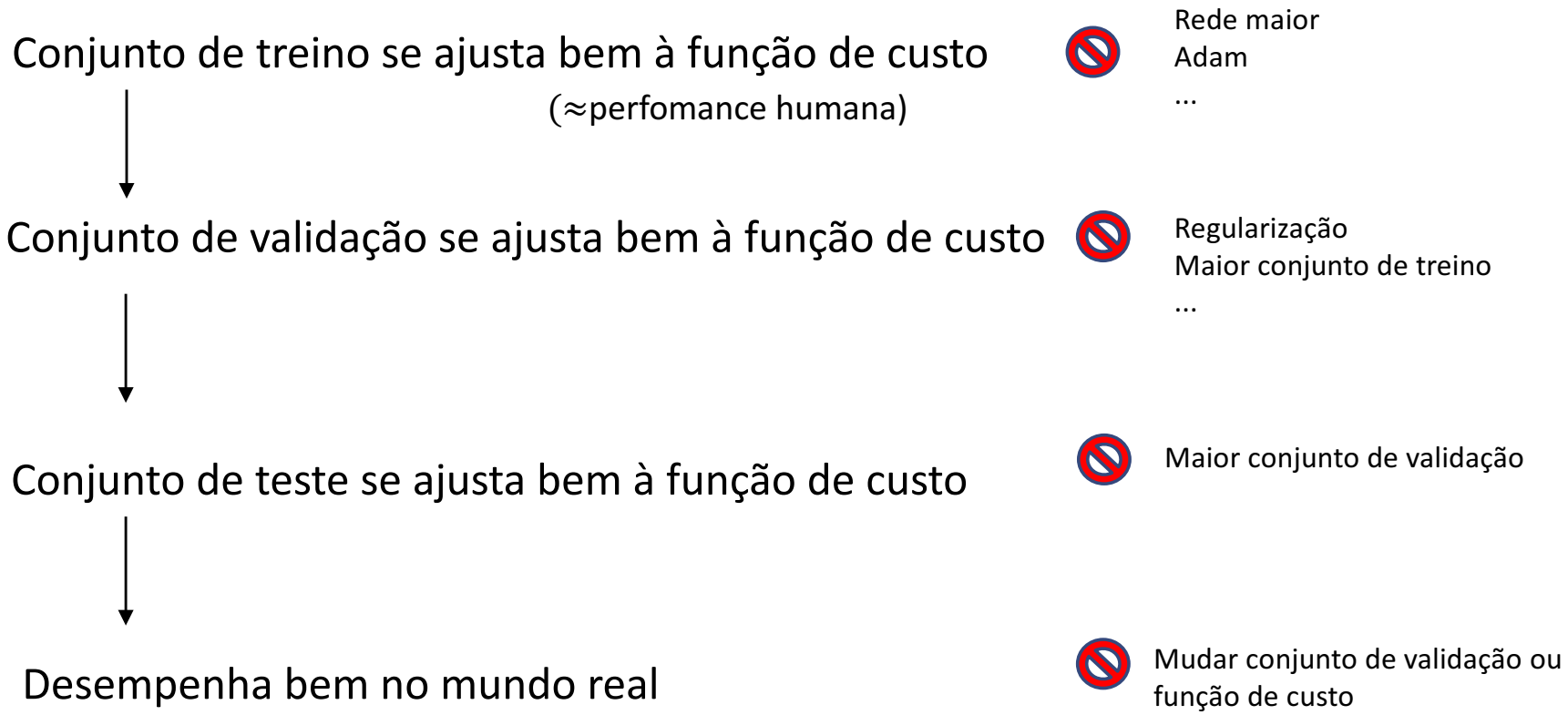
Receita básica para ajustar modelo



”Bias variance tradeoff”



Cadeia de validações em ML



Resumo

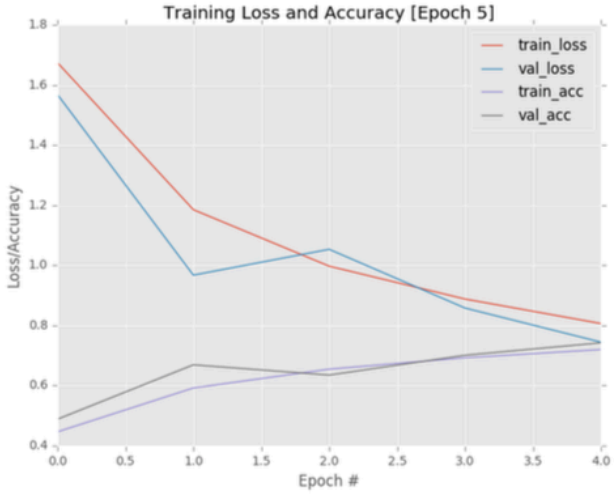
- Ideias
  - Coletar mais dados
  - Adquirir conjunto de treino mais diverso
  - Treinar o algoritmo por mais tempo
  - Tentar Adam ao invés de Gradiente descendente
  - Tentar rede maior
  - Tentar rede menor
  - Adicionar dropout
  - Adicionar Regularização L2
  - Ajustar arquitetura da rede
    - Função de ativação
    - # camadas escondidas
    - Taxa de aprendizado
    - ...

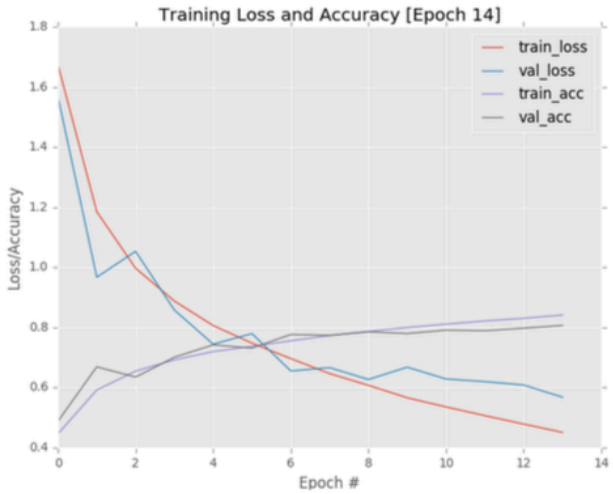
Estudo de Caso 1

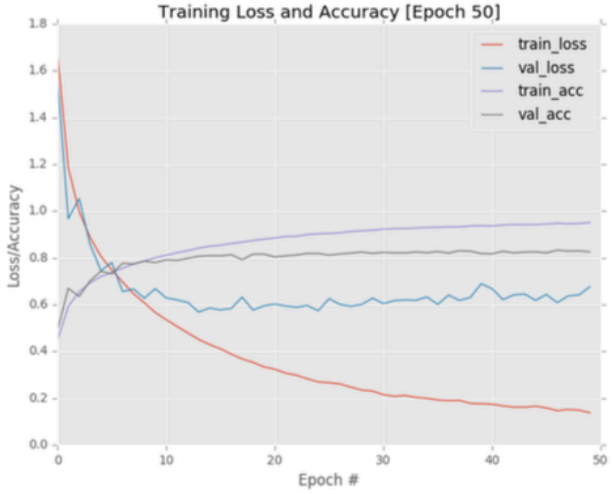
- Esse problema é adaptado de uma aplicação real

- Estudo de Caso1.ipynb

Monitorando o processo de treino







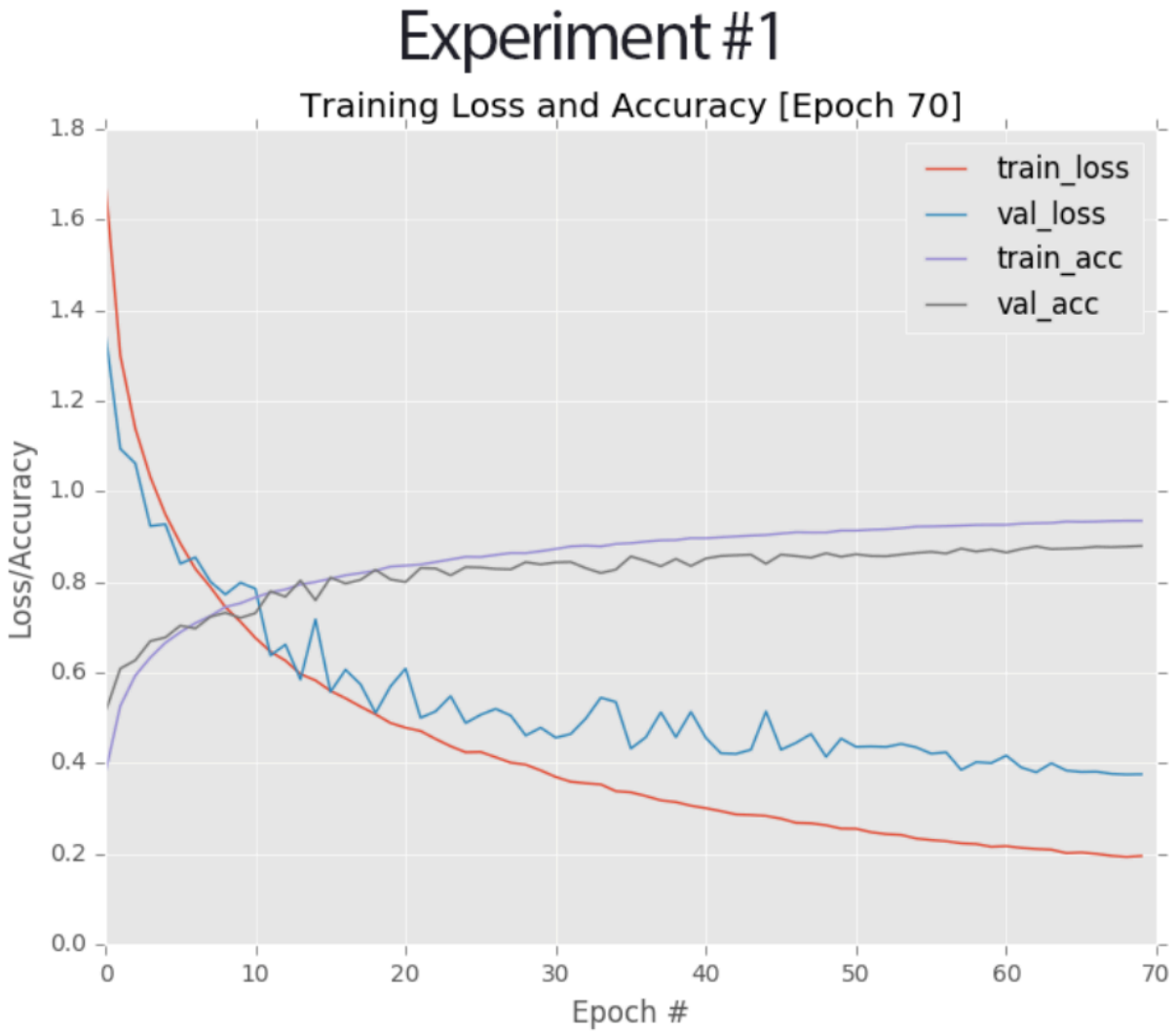
Preste atenção na curva de treinamento

- Você está aplicando alguma regularização?
- Sua taxa de aprendizado está alta?
- Sua rede é muito profunda?

MiniGoogleNet - Experimentos

- Experimento 1
  - Lr = 0.001

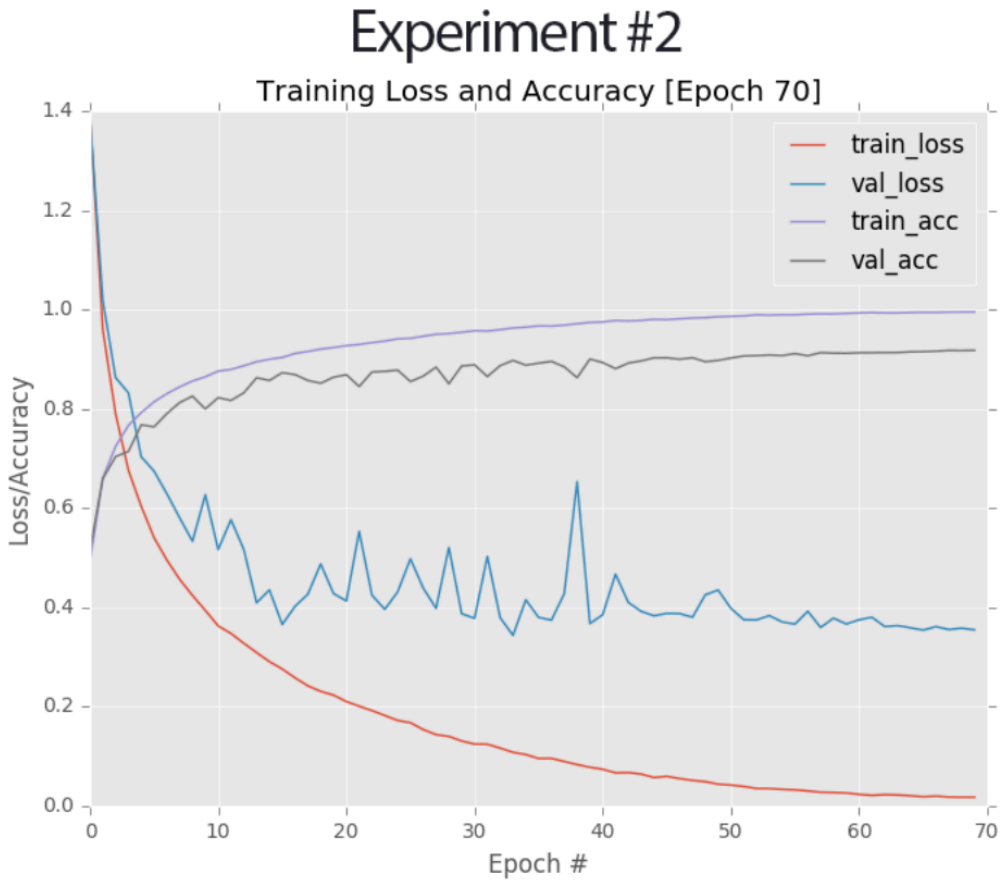
Acurácia = **87,5%**



MiniGoogleNet - Experimentos

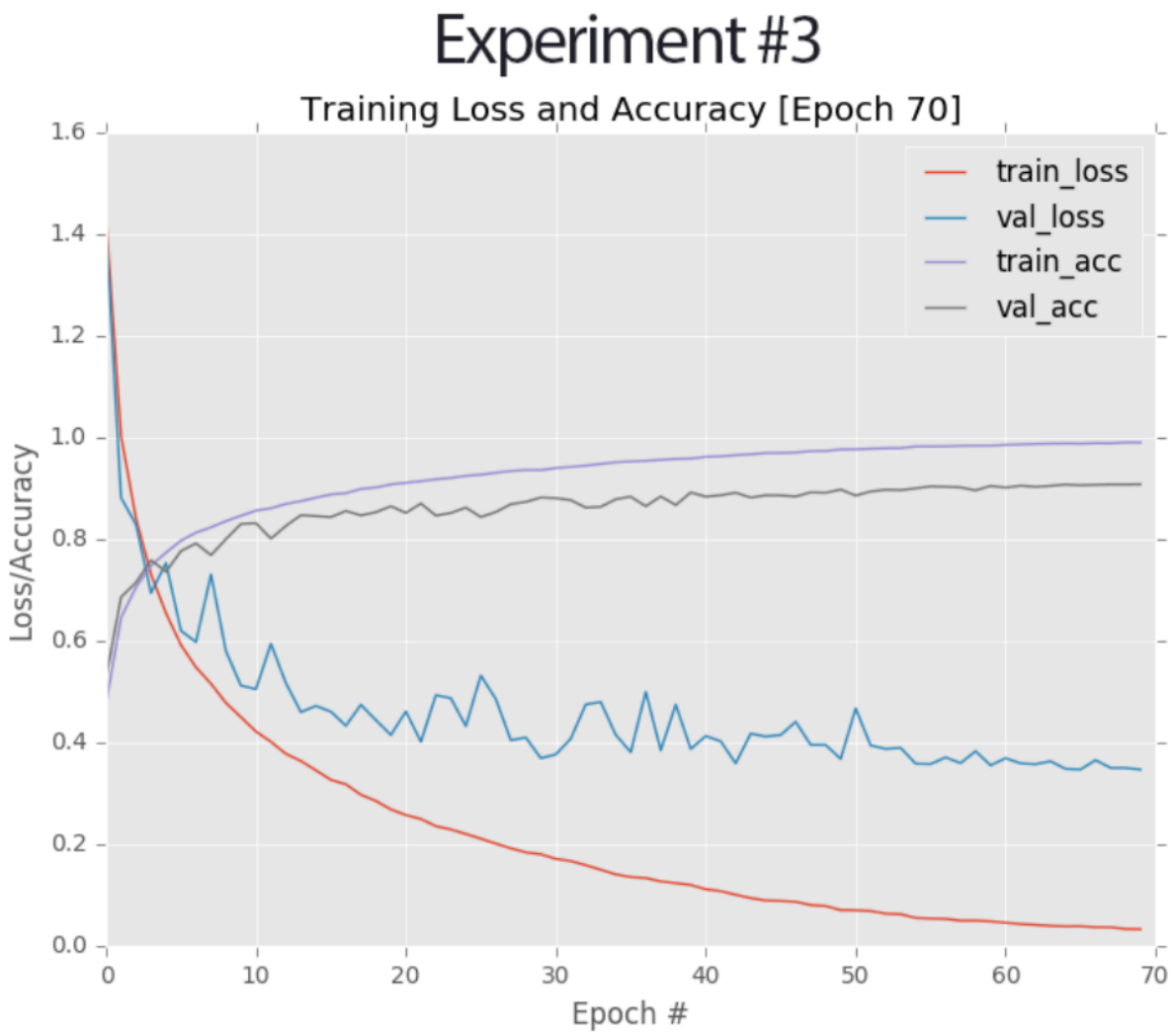
- Experimento 2
  - Lr = 0.001-> 0.01

Acurácia = **91,79%**



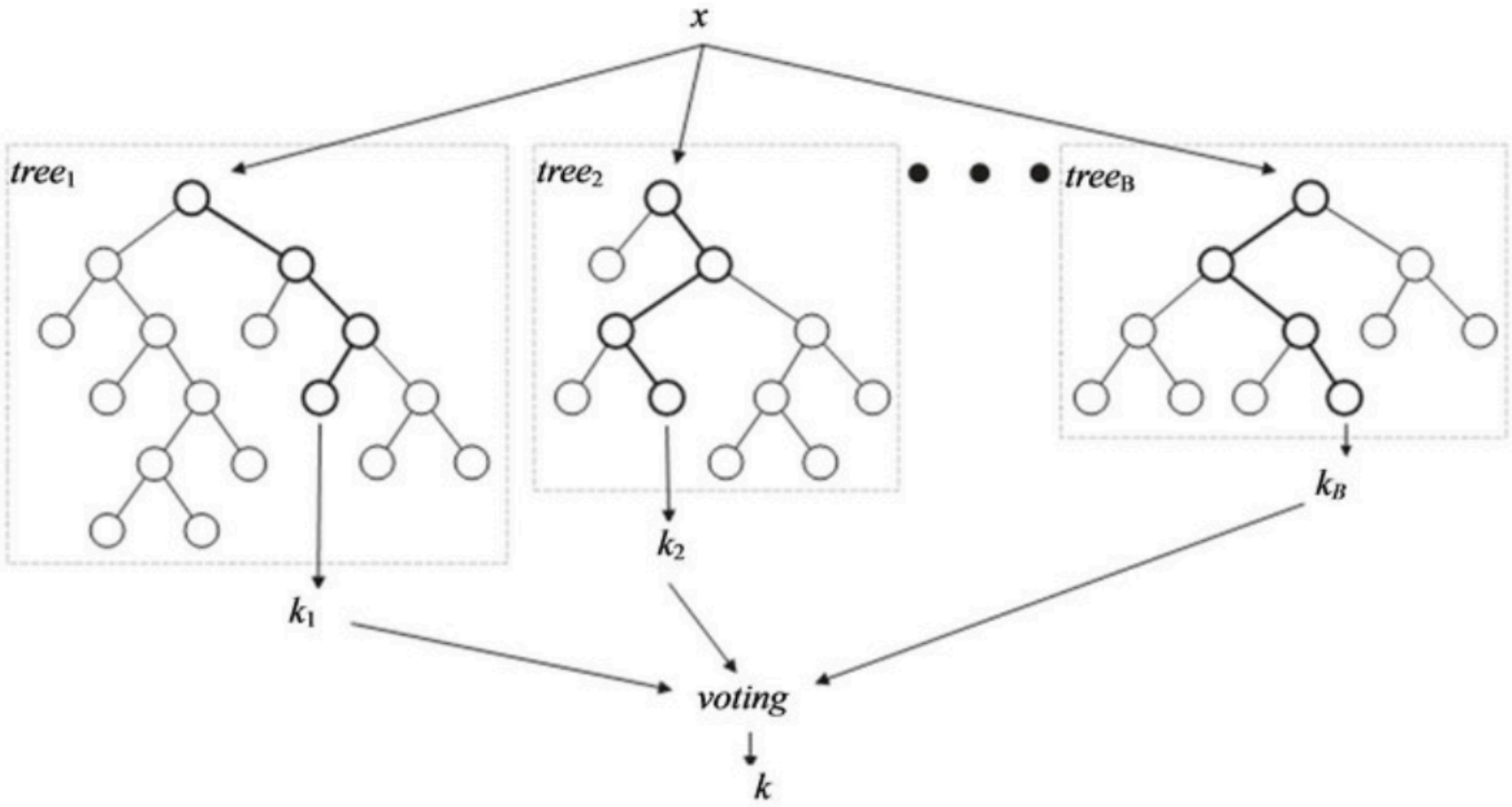
MiniGoogleNet - Experimentos

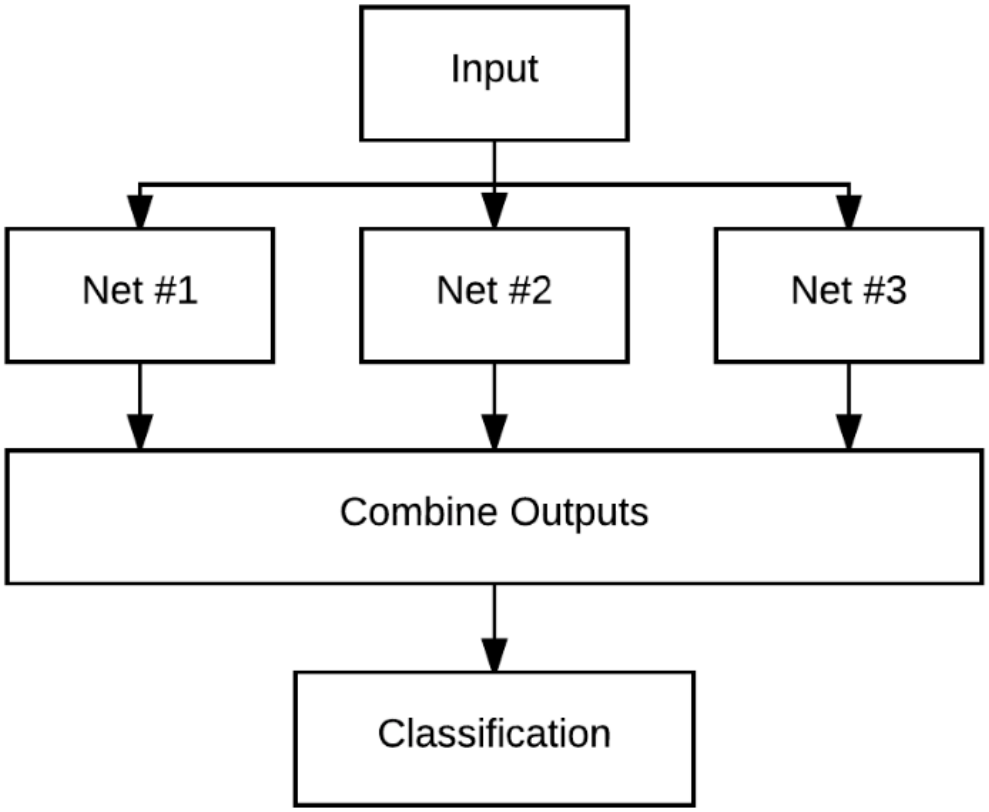
- Experimento 3
  - Lr = 0.001-> 0.01 ->0,005 Acurácia = **90,81%**



Método Ctrl+c – Resnet + cifar10

Ensembles


x
tree1
tree2
treeB
k1
k2
kB
voting
k


Input
Net #1
Net #2
Net #3
Combine Outputs
Classification


IDEAS
Think?
good
click
Inspiration
CHECK LIST:
- SEO
- 3G / 4G
cloud
SHARE