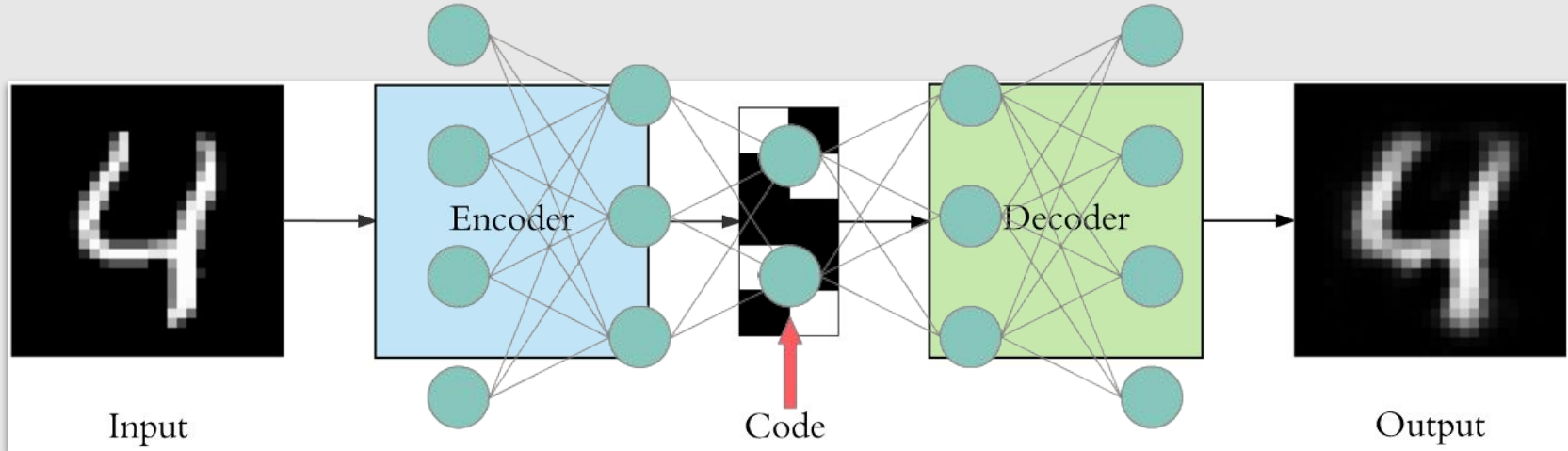


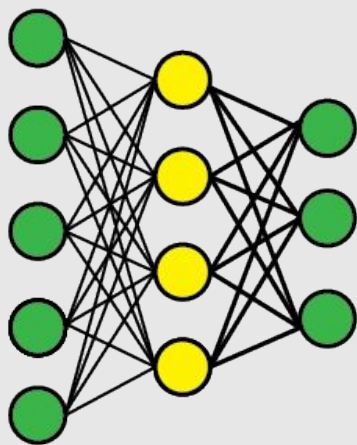
Maior Dúvida da Aula

1. Treinar uma rede em duas sessões de 50 épocas é a mesma coisa que treinar a rede em uma única sessão de 100 épocas?
2. Posso afirmar que em uma rede neural a regularização reduz o overfitting tornando a rede menos densa (já que alguns pesos seriam “zerados”)? Se sim, posso associar densidade com complexidade da rede?
3. Foi comentado bastante sobre como as funções de ativação foram estudadas para auxiliar no problema de vanishing gradient. Houve alguma abordagem em relação ao problema de exploding gradient?
4. Sempre vejo explicações que modelagens com ANN buscam gerar representações esparsas. Porém não consegui entender a intuição por trás disso, seria para facilitar a classificação/regressão feita pela rede? Por que?

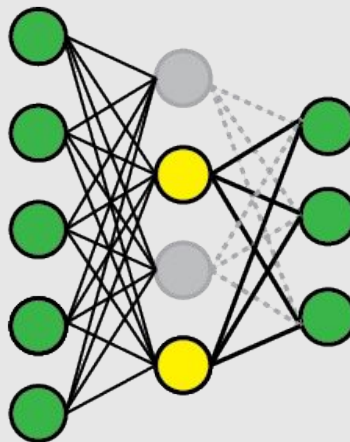
5. No início da aula conversamos sobre uma arquitetura utilizada em aprendizado não supervisionado onde pegamos o modelo com uma representação compactada da entrada. Esse modelo não seria um modelo overfitado?



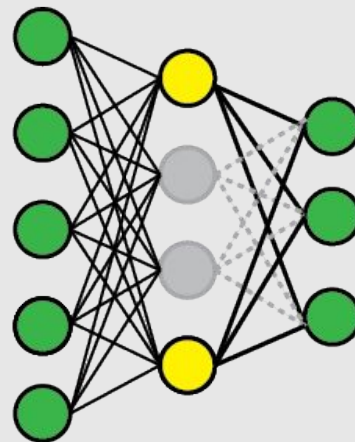
6. Existe alguma técnica para “ignorar” algumas camadas aleatoriamente durante o treinamento (feedforward)? A ideia seria analisar o efeito da remoção de uma ou mais camadas durante o treino.



Standard network



After applying dropout



“Dropout: A Simple Way to Prevent Neural Networks from Overfitting”, 2012, <https://www.cs.toronto.edu/~hinton/absps/JMLRdropout.pdf>

7. Dependendo da função de ativação é importante também normalizar as features? Se seleciono a função Tanh, as features tem que estar entre -1 e 1?

Batch Normalization

To increase the stability of a neural network, batch normalization **normalizes the output** of a previous activation layer **by subtracting the batch mean** and **dividing by the batch standard deviation**.

“Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”,
ICML 2015, <https://arxiv.org/pdf/1502.03167>

“Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”, <https://arxiv.org/pdf/1502.03167>

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_1 \dots x_m\}$;

Parameters to be learned: γ, β

Output: $\{y_i = \text{BN}_{\gamma, \beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m} \sum_{i=1}^m x_i \quad // \text{ mini-batch mean}$$

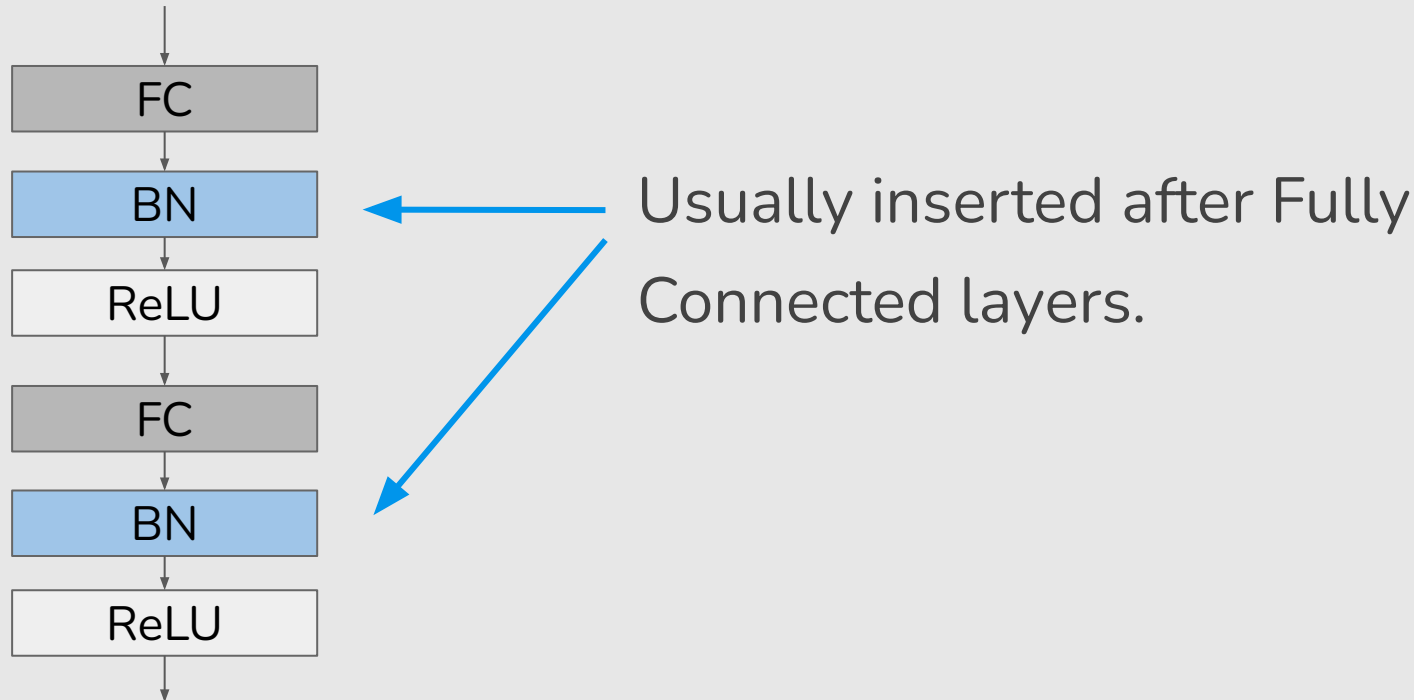
$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m} \sum_{i=1}^m (x_i - \mu_{\mathcal{B}})^2 \quad // \text{ mini-batch variance}$$

$$\hat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \quad // \text{ normalize}$$

$$y_i \leftarrow \gamma \hat{x}_i + \beta \equiv \text{BN}_{\gamma, \beta}(x_i) \quad // \text{ scale and shift}$$

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

Batch Normalization



Batch Normalization: An Example (MNIST)



<http://yann.lecun.com/exdb/mnist/>

Batch Normalization: An Example (MNIST)

Model **without** batch normalization:

```
from tensorflow.keras import layers

# Creating the model
model_without_bn = tf.keras.Sequential()

# Architecture
model_without_bn.add(layers.Dense(256, activation='relu', input_shape=(784,)))
model_without_bn.add(layers.Dense(128, activation='relu'))
model_without_bn.add(layers.Dense(64, activation='relu'))
model_without_bn.add(layers.Dense(10, activation='softmax'))
```

Model **with** batch normalization:

```
from tensorflow.keras import layers

# Creating the model
model_without_bn = tf.keras.Sequential()

# Architecture
model_with_bn.add(layers.Dense(256, use_bias=False, input_shape=(784,)))
model_with_bn.add(layers.BatchNormalization())
model_with_bn.add(layers.Activation('relu'))

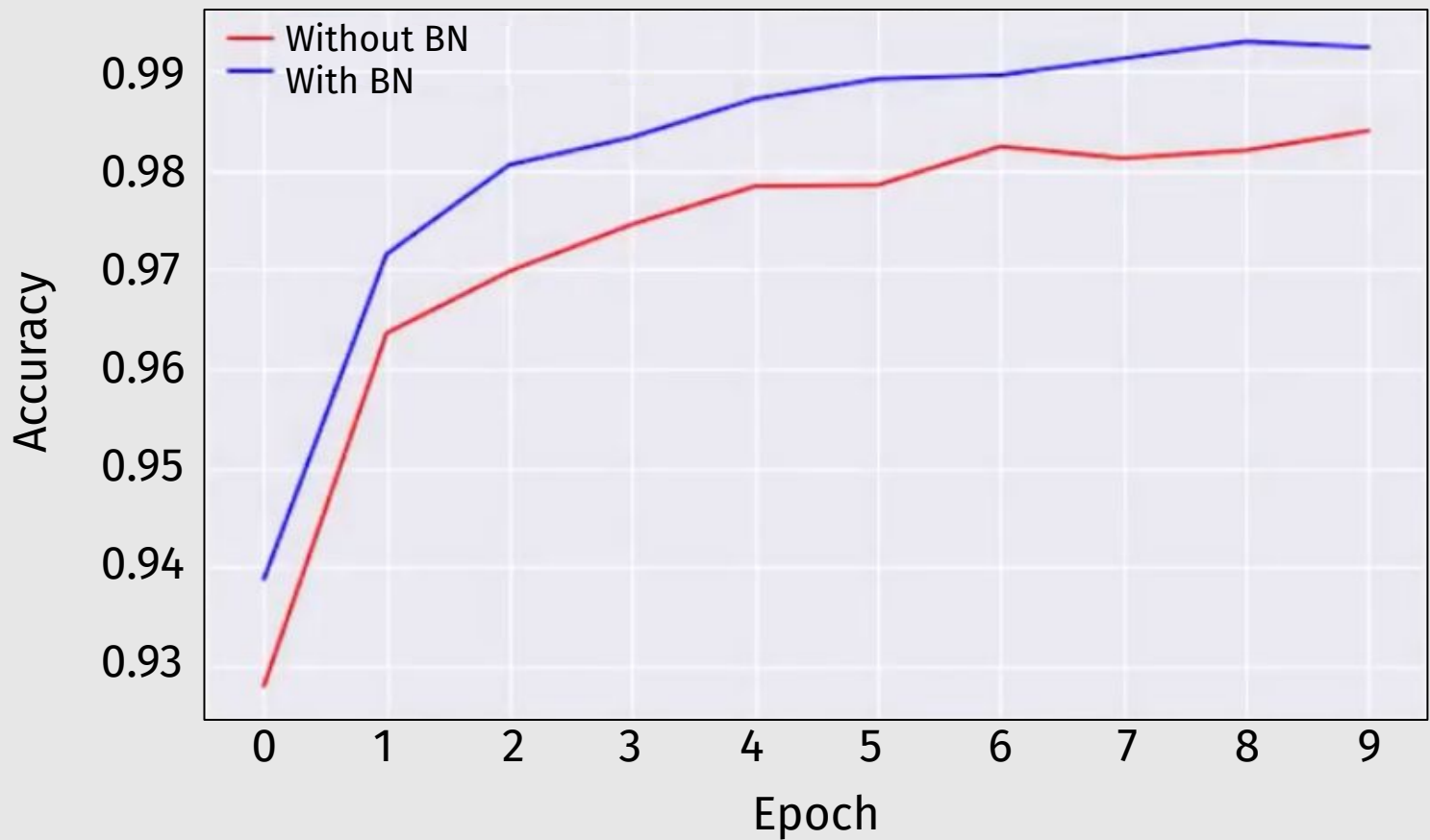
model_with_bn.add(layers.Dense(128, use_bias=False))
model_with_bn.add(layers.BatchNormalization())
model_with_bn.add(layers.Activation('relu'))

model_with_bn.add(layers.Dense(64, use_bias=False))
model_with_bn.add(layers.BatchNormalization())
model_with_bn.add(layers.Activation('relu'))

model_with_bn.add(layers.Dense(10, activation='softmax'))
```

Batch Normalization (1): An Example (MNIST)

- epochs = 10
- batch size = 128
- learning rate = 0.01
- data normalization: $X/255$
- weight init: glorot_uniform

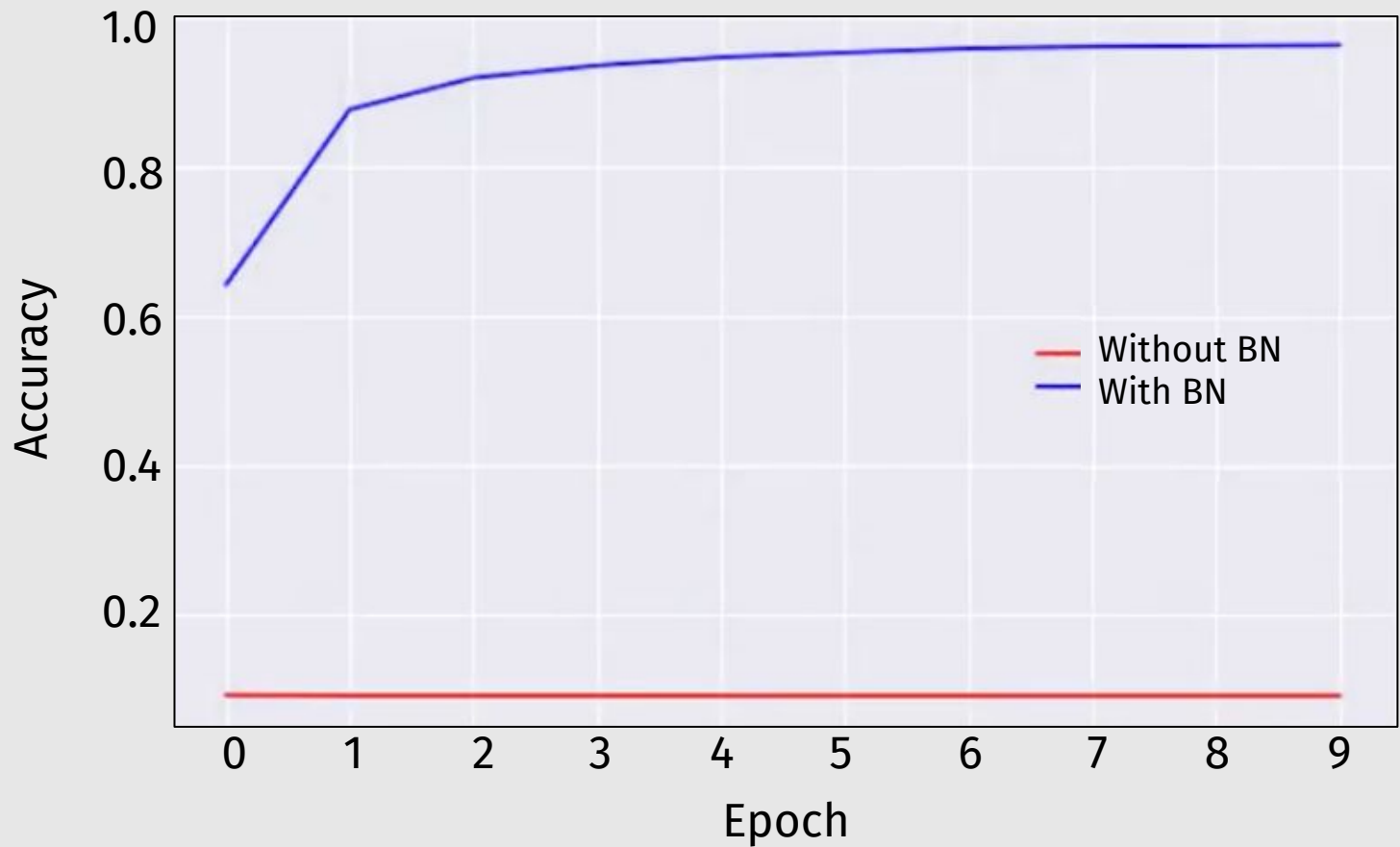


Without Batch Normalization test-acc: 0.966

With Batch Normalization test-acc: 0.974

Batch Normalization (2): An Example (MNIST)

- epochs = 10
- batch size = 128 \rightarrow 1024
- learning rate = 0.01 \rightarrow 1
- data normalization: $X/255$
- weight init: glorot_uniform

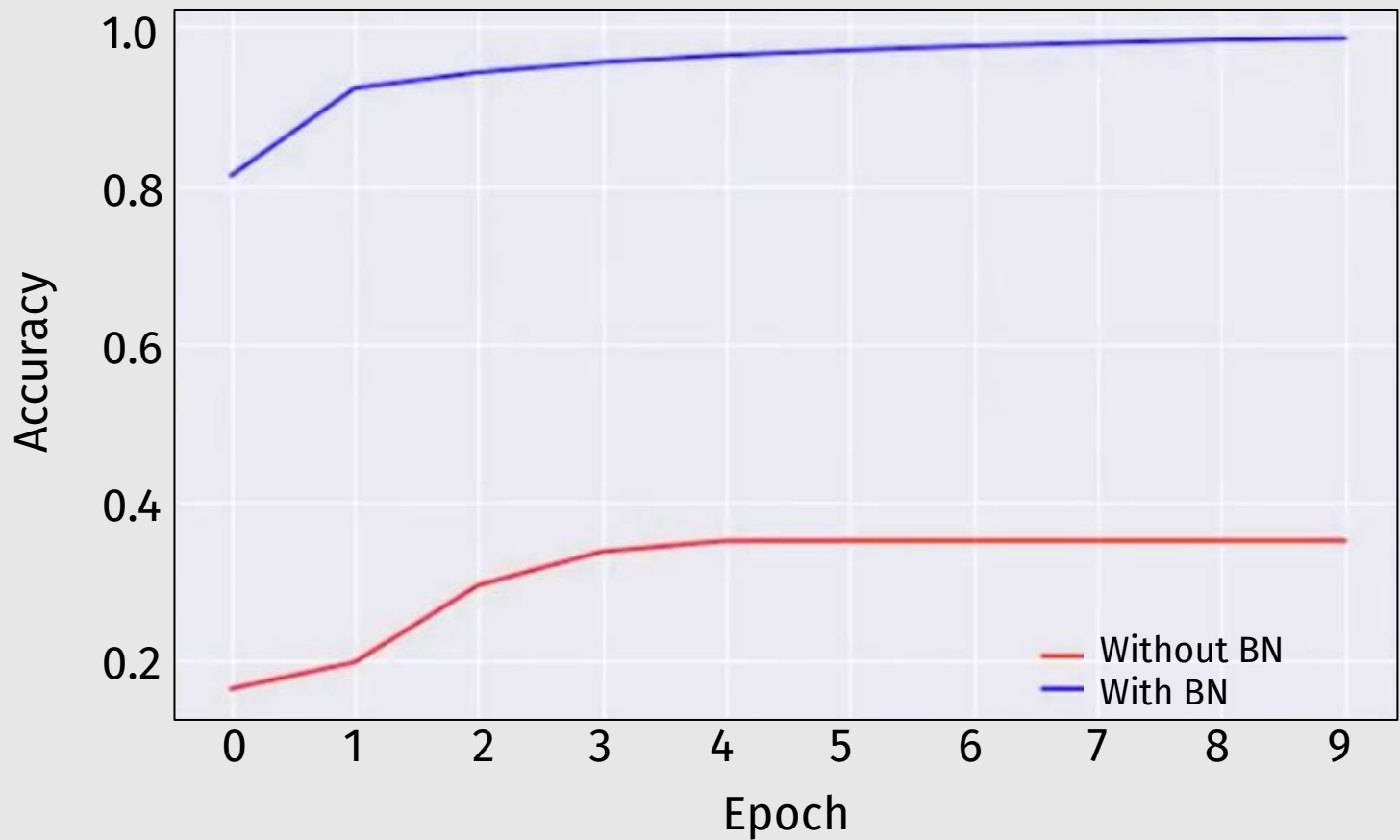


Without Batch Normalization test-acc: 0.089

With Batch Normalization test-acc: 0.950

Batch Normalization (3): An Example (MNIST)

- epochs = 10
- batch size = 128
- learning rate = 0.01
- data normalization: $X/255$
- weight init: glorot_uniform →
RandomUniform(minval=-5, maximal=5)

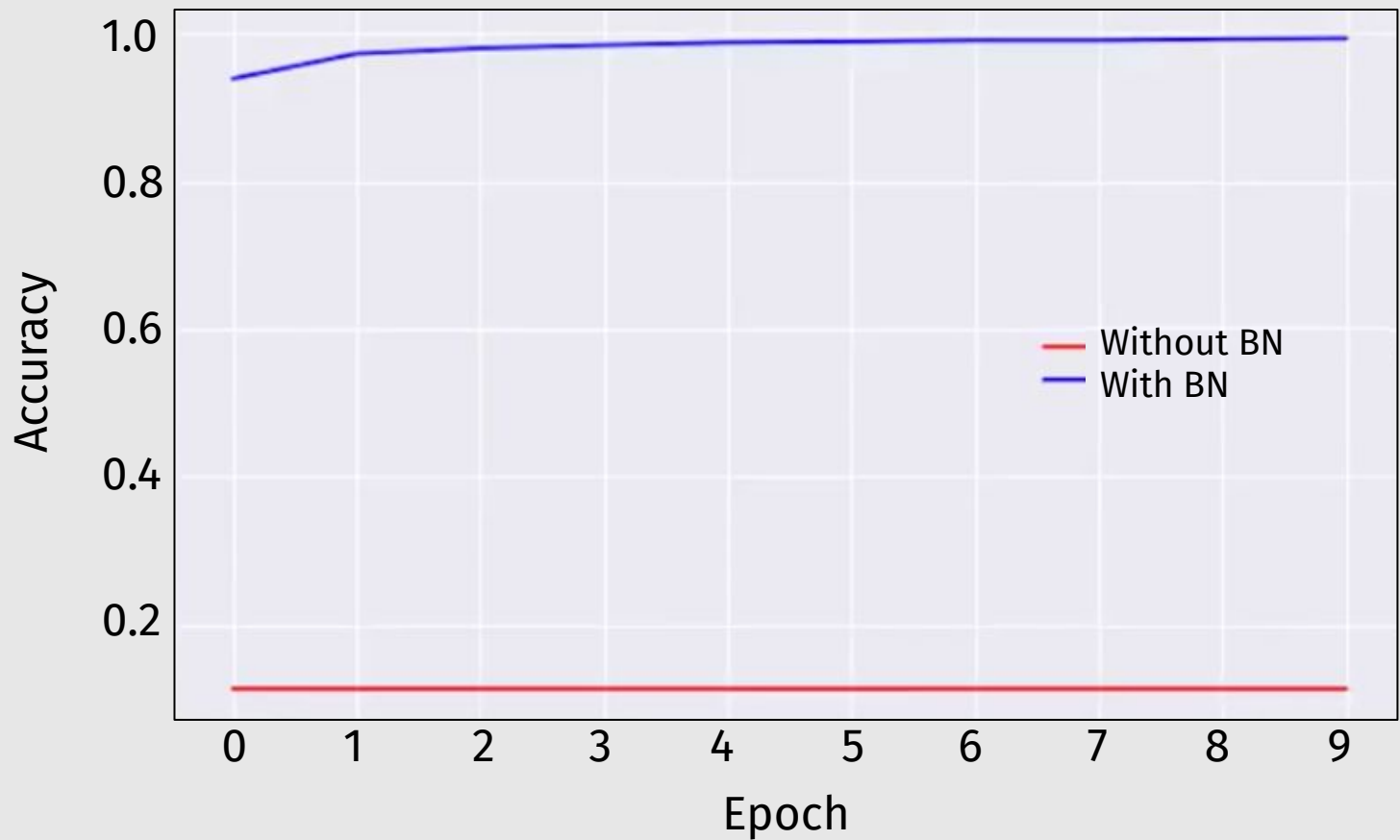


Without Batch Normalization test-acc: 0.352

With Batch Normalization test-acc: 0.966

Batch Normalization (4): An Example (MNIST)

- epochs = 10
- batch size = 128
- learning rate = 0.01
- data normalization: $X/255 \rightarrow$ no norm
- weight init: glorot_uniform



Without Batch Normalization test-acc: 0.113

With Batch Normalization test-acc: 0.977

Batch Normalization

- Makes deep networks much easier to train!
- Improves gradient flow
- Faster convergence
- Networks become more robust to initialization
- Acts as regularization during training

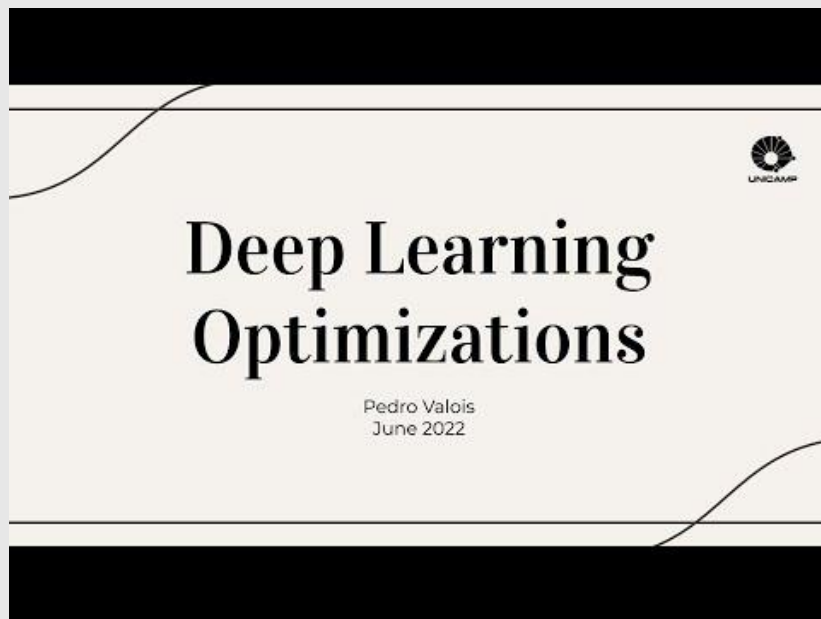
“Fitting Batch Norm Into Neural Networks”, deeplearning.ai <https://youtu.be/em6dfRxYkYU>

“How does Batch Normalization Help Optimization?”, Ilyas et al., NeurIPS 2018, <http://gradientscience.org/batchnorm/>

8. Para realizar o Backpropagation podemos utilizar qualquer tipo de função de ativação?
9. Achei muito interessante a playlist de vídeos do 3Blue1Brown! Nos vídeos é possível visualizar certinho como as contas são feitas e a rede é modificada conforme as iterações de treinamento.
10. É possível retropropagar o erro apenas para uma “parte” da rede?
11. Ainda estou com um pouco de dúvida com relação ao bias. A definição dele é empírica? Ele é o mesmo bias que usamos em outras regressões, por exemplo a linear? E por que utilizar um em cada camada, ao invés de colocá-lo apenas na última?

12. Temos que fazer o cálculo do backpropagation toda vez que fizer um modelo? Se não, em quais momentos é aconselhável fazer essa conta para ver o comportamento?
13. Por que em redes neurais maiores, a função ReLU tende a ter um desempenho melhor do que a logística?
14. Quando vamos fazer o update de um peso em determinada camada, a gente usa o delta resultante do erro na rede toda, ou aquele calculado naquela camada em específico?
15. Durante o treinamento é possível saber, em uma situação de inconsistência numérica (custo indo para infinito ou dando NAN), se o problema vem dos valores do gradiente? Isto é, se é causado pelo desaparecimento ou por valores demasiados altos dele?

16. De um modo geral, podemos dizer que a etapa mais custosa das iterações de treino de uma rede neural é a do backpropagation? Como é possível otimizá-la?



<https://youtu.be/dbOMVC5huNA>

Train longer, generalize better: closing the generalization gap in large batch training of neural networks, 2018

Low-memory neural network training: A technical report, 2019

An empirical model of large-batch training, 2018

Large batch training of convolutional networks, 2018

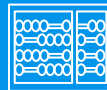
Mixed precision training, NVIDIA, 2017

Context encoding for semantic segmentation, 2018

<https://www.techspot.com/article/2049-what-are-ten-sor-cores>



recod.ai
reasoning for complex data



Artificial Neural Networks

Machine Learning

Prof. Sandra Avila

Institute of Computing (IC/Unicamp)

MC886/MO444, September 29, 2022

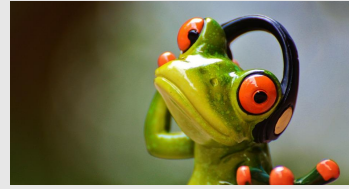
Multi-class Classification



Cat



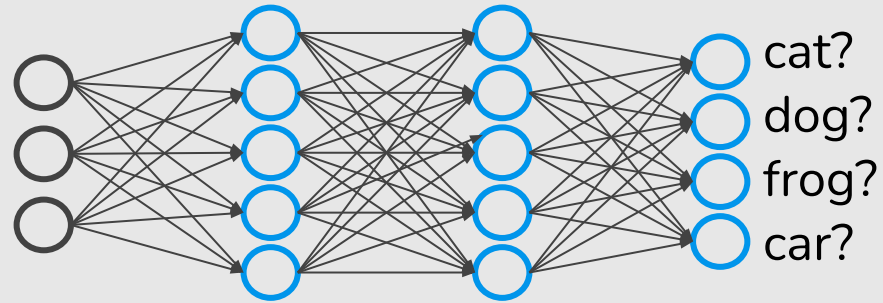
Dog



Frog



Car





Cat



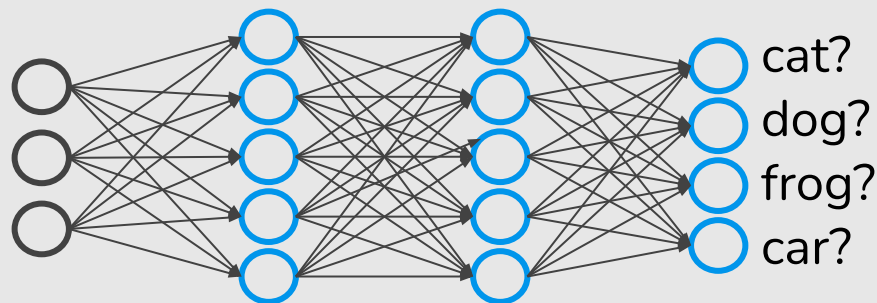
Dog



Frog



Car



Want $h_{\Theta}(x) \approx \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$, when cat

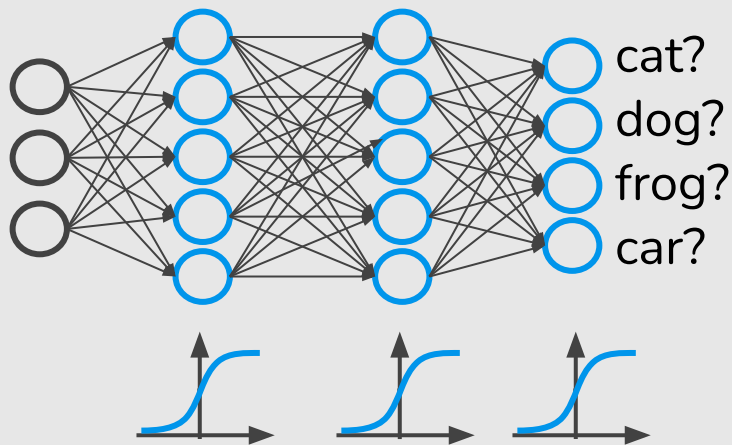
$h_{\Theta}(x) \approx \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$, when dog

$h_{\Theta}(x) \approx \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$, when frog

$h_{\Theta}(x) \approx \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$, when car

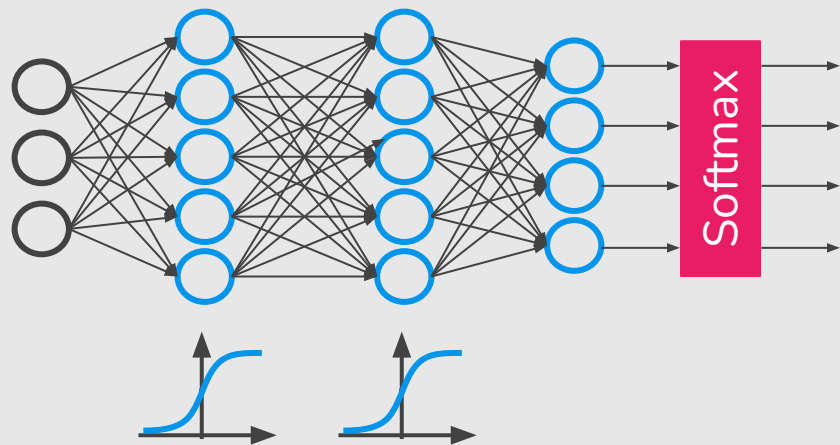
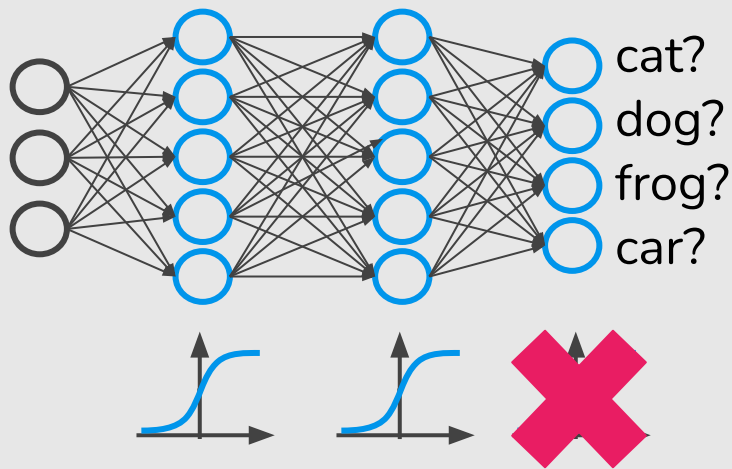
Softmax Classification

The **output layer** is typically modified **by replacing** the individual activation functions **by a shared softmax** function.



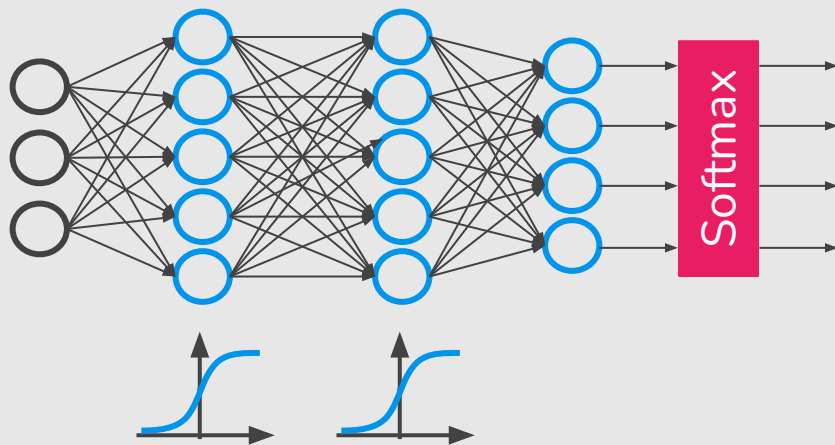
Softmax Classification

The **output layer** is typically modified **by replacing** the individual activation functions **by a shared softmax** function.



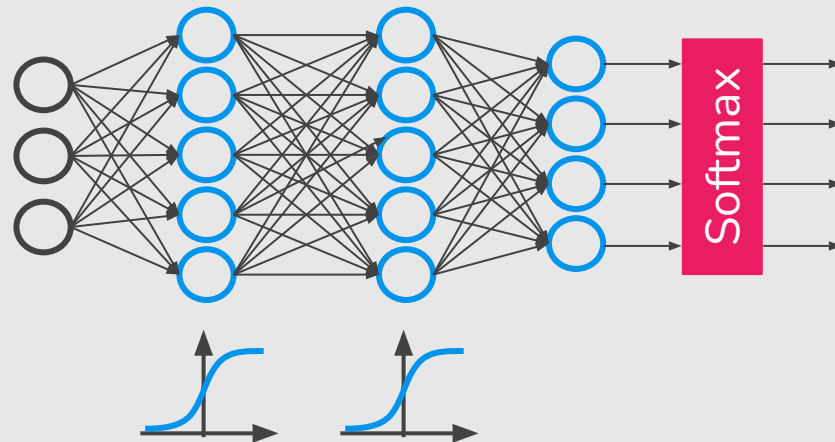
Softmax Classification

The **output layer** is typically modified **by replacing** the individual activation functions **by a shared softmax** function.



$$f(\mathbf{z})_k = \frac{e^{z_k}}{\sum_{j=1}^K e^{z_j}}$$


Softmax Classification






Cat	5.1		164.0		0.87
Dog	3.2	➡	24.5	➡	0.13
Frog	-1.7		0.18		0.00
Car	-2.0		0.13		0.00


$$f(\mathbf{z})_k = \frac{e^{z_k}}{\sum_{j=1}^K e^{z_j}}$$


Neural Networks (3Blue1Brown)




Search





 Home


 Trending


 Subscriptions


LIBRARY

 History


 Watch later

 Liked videos

 Chansons Franca...

 Show more

SUBSCRIPTIONS

 Instituto de Comp...



Neural Networks

PLAY ALL

Neural networks

4 videos • 320,262 views • Last updated on Aug 1, 2018



3Blue1Brown SUBSCRIBED 1.1M 

SEASON 3 ▾

1



Neural Networks

19:13

3BLUE1BROWN SERIES S3 • E1

But what *is* a Neural Network? | Deep learning, chapter 1

3Blue1Brown

2



How machines learn

21:01

3BLUE1BROWN SERIES S3 • E2

Gradient descent, how neural networks learn | Deep learning, chapter 2

3Blue1Brown

3



Backpropagation

13:54

3BLUE1BROWN SERIES S3 • E3

What is backpropagation really doing? | Deep learning, chapter 3

3Blue1Brown

4



Backpropagation calculus

10:18

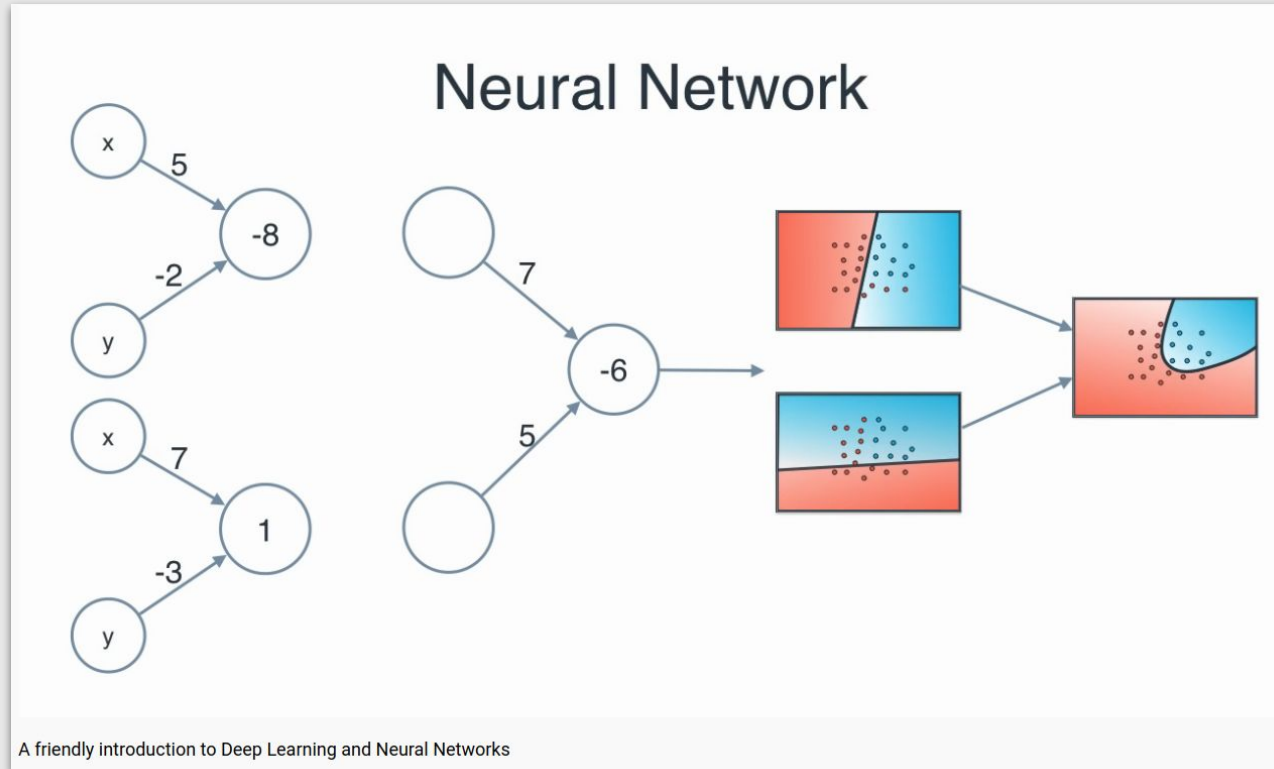
3BLUE1BROWN SERIES S3 • E4

Backpropagation calculus | Deep learning, chapter 4

3Blue1Brown

“A friendly introduction to Neural Networks”

<https://youtu.be/BR9h47Jtqyw>



Thumbnail Classification - Redes Neurais com PyTorch.ipynb ☆

File Edit View Insert Runtime Tools Help All changes saved

00:00:12:19



+ Code + Text

```
[ ] 25 train_size = int(0.75*len(data))
    26 idx = torch.randperm(len(data))
    27 train_sampler = SubsetRandomSampler(idx[0:train_size])
    28 test_sampler = SubsetRandomSampler(idx[train_size:])
    29
    30 train_loader = DataLoader(data, sampler=train_sampler,
    31                           batch_size=10, num_workers=4)
    32
    33 test_loader = DataLoader(data, sampler=test_sampler,
    34                           batch_size=10, num_workers=4)
```

Rede Neural em PyTorch

<https://youtu.be/Vfzm1-cfLuc>

Imports de biblioteca e verificação de dispositivo de hardware.

```
1 import torch
2 if
```

cuda

Implementando a arquitetura

```
[ ] 1
```

Redes Neurais em PyTorch | Programando em 10 minutos #1

17,671 views • Feb 19, 2020

1.7K 19 SHARE SAVE ...



Peixe Babel
84.4K subscribers

SEE PERKS

SUBSCRIBED



**THIS IS A NEURAL
NETWORK.**

**IT MAKES MISTAKES.
IT LEARNS FROM THEM.**

**BE LIKE A NEURAL
NETWORK.**

