

# MACHINE LEARNING

Uma introdução prática





# HELLO WORLD!

## **Autores**

@colombelli

fcolombelli@inf.ufrgs.br

@jpelax

jprodrigues@inf.ufrgs.br

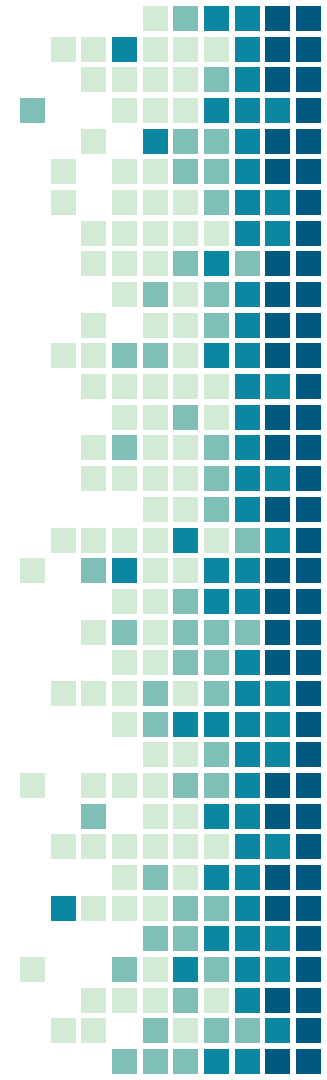
## **Monitores**

@birromer

bhflores@inf.ufrgs.br

@phpgit2

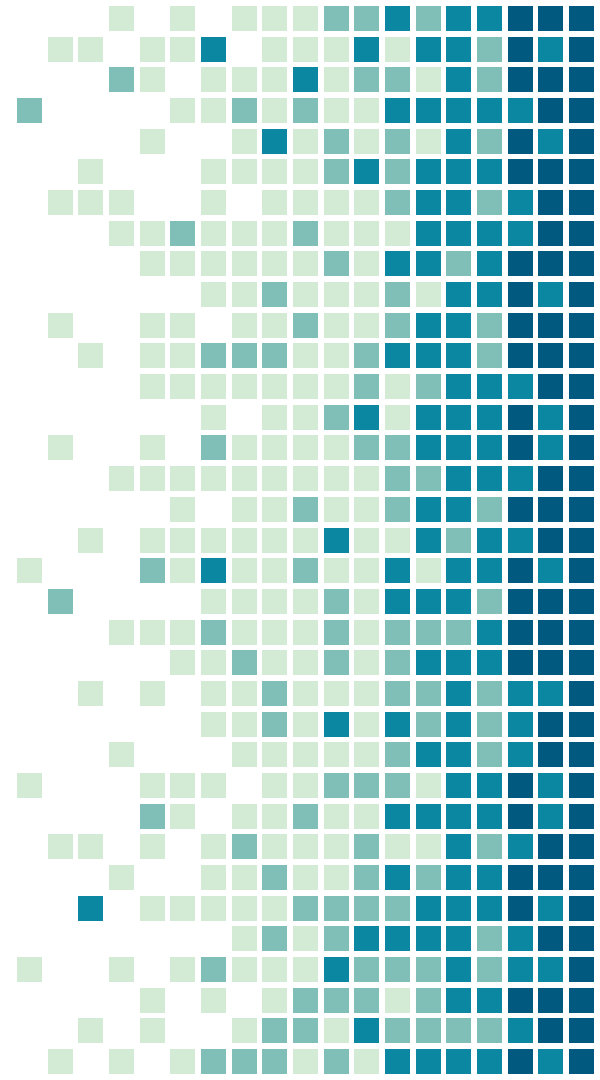
pedro.fiorentin@inf.ufrgs.br



2.

# Enfoque do curso

Redes neurais artificiais na prática



# TECNOLOGIAS E FERRAMENTAS

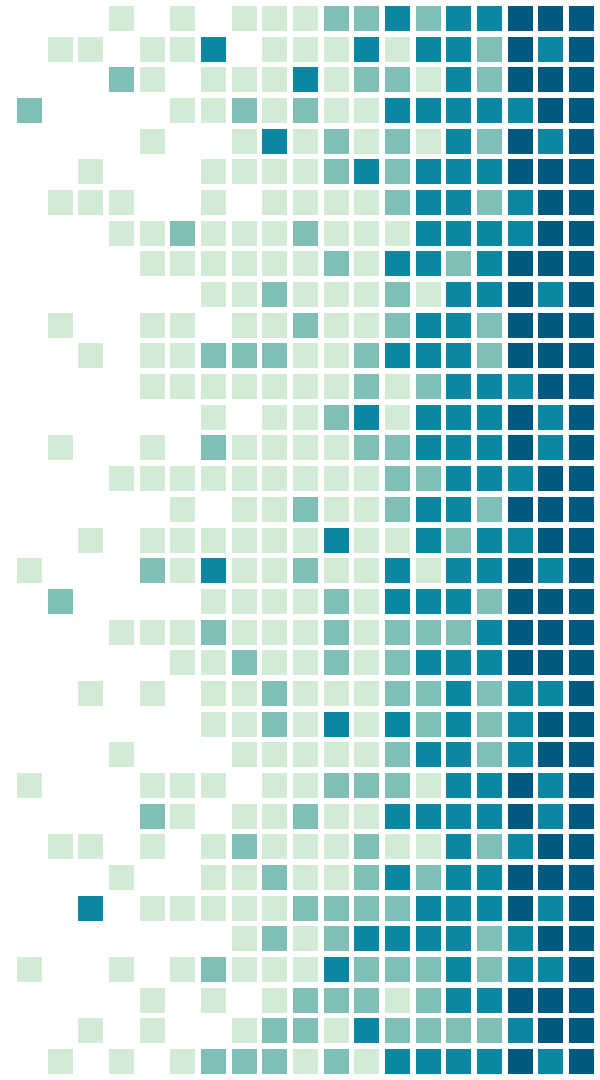
- Python + Google Colab
- Keras
- TensorFlow
- PyTorch
- Scikit-Learning



3.

# Motivação

Por que estudar machine learning?

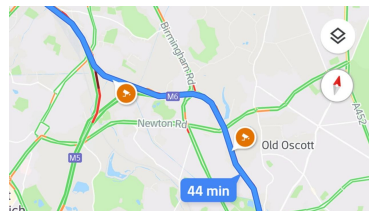


# Para quê?

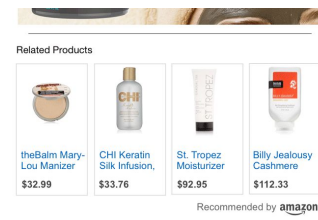
## Robótica



## Transporte



## Varejo



## Governo



## Negócios



## Saúde



# Data doesn't lie

- "In the past year, the number of PhD graduates on LinkedIn who say they have AI expertise has risen by 66%" - *Nature.com*
- "Machine learning engineer is the best job of 2019 due to growing demand and high salaries" - *Indeed.com*
- "Global machine learning market is expected to grow from \$1.4B in 2017 to \$8.8B by 2022" - *ResearchAndMarkets.com*



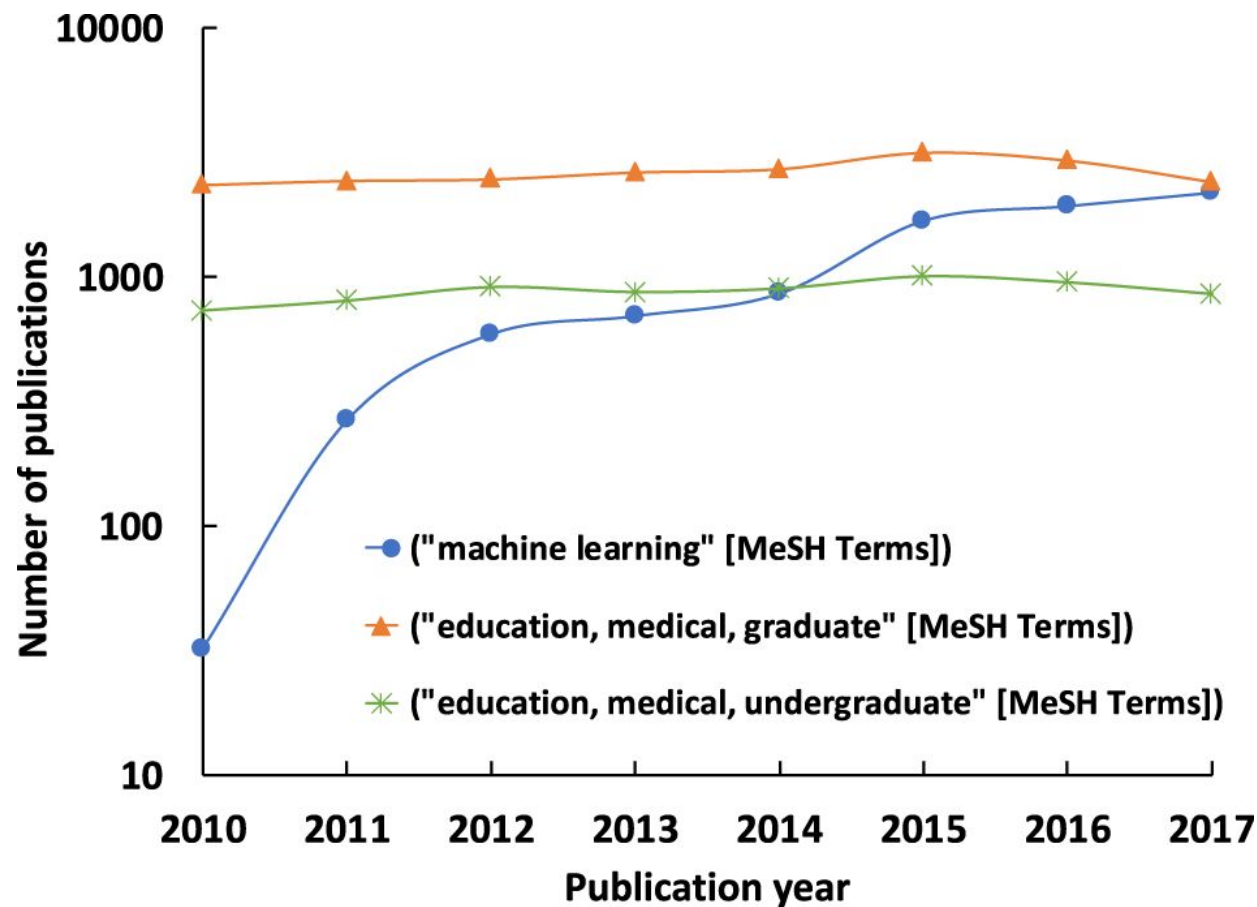
## Indeed's best jobs of 2019

Rank	Job title	% growth in # of postings, 2015–2018	Average base salary	Job title's # of postings per 1million total tobs, 2018
1	Machine Learning Engineer	344%	\$146,085	179
2	Insurance Broker	242%	\$86,498	32
3	Full-stack Developer	206%	\$114,316	828
4	Insurance Advisor	190%	\$81,479	45
5	Litigation Attorney	168%	\$101,289	92
6	Litigation Associate	165%	\$98,982	53
7	Dental Hygienist	157%	\$78,110	878
8	Associate Attorney	149%	\$75,515	281
9	Realtor	138%	\$96,820	221
10	Salesforce Developer	129%	\$112,031	170

Fonte: <http://blog.indeed.com/2019/03/14/best-jobs-2019/>

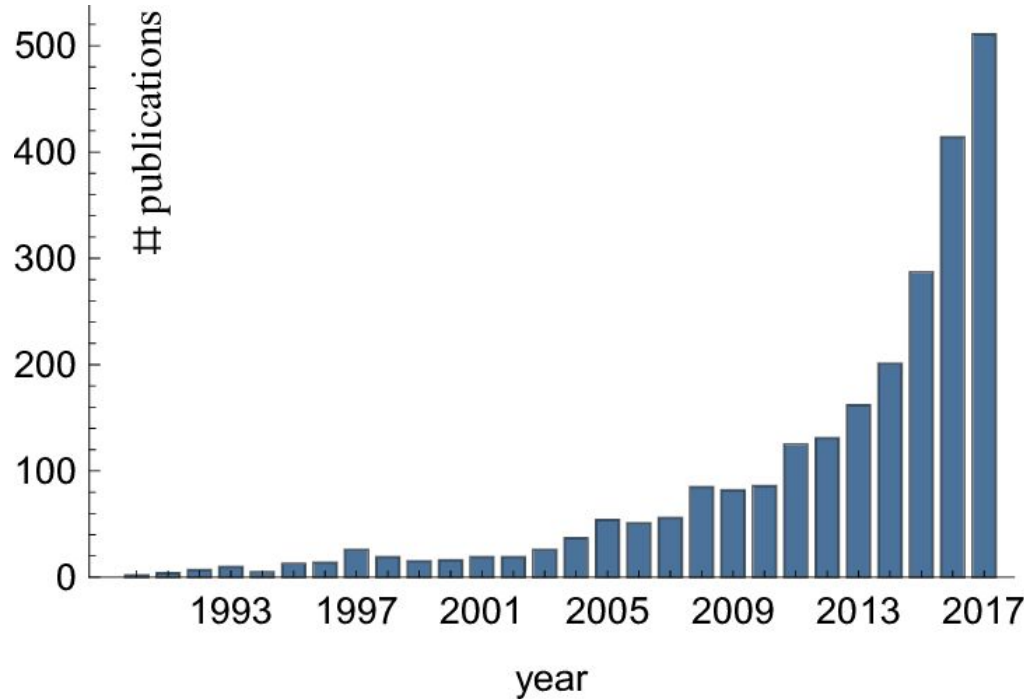






Fonte: <https://www.nature.com/articles/s41746-018-0061-1>

Number of publications per year from a web of science search for articles with topics of machine learning and either chemistry or materials.



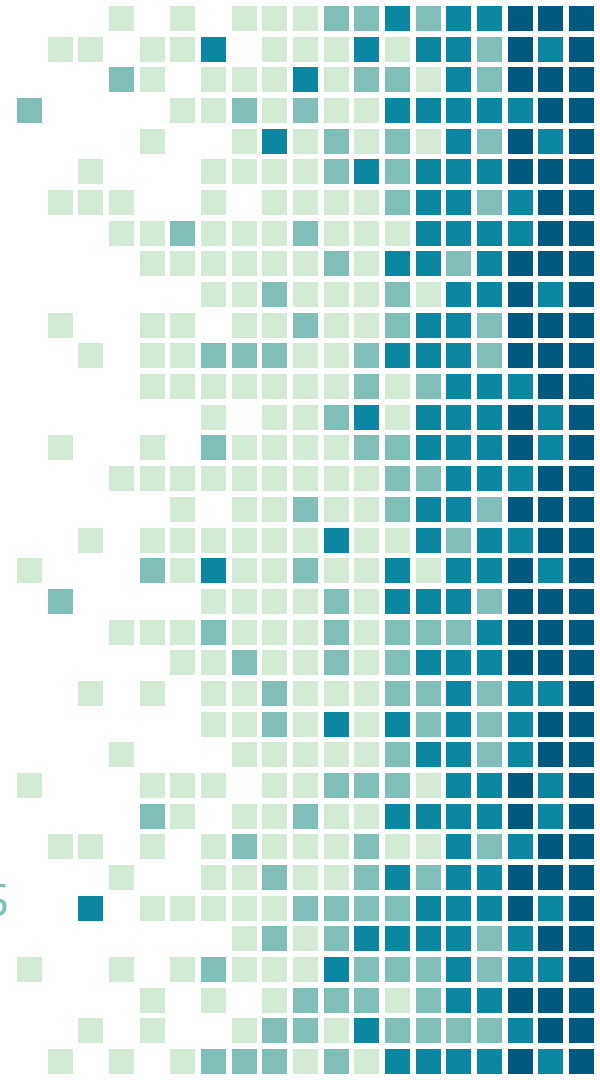
Fonte: [https://www.researchgate.net/figure/Number-of-publications-per-year-from-a-web-of-science-search-for-articles-with-topics-of\\_fig1\\_326028833](https://www.researchgate.net/figure/Number-of-publications-per-year-from-a-web-of-science-search-for-articles-with-topics-of_fig1_326028833)



4.

# Conceitos envolvidos

Situando-se nesse universo de informações



# Artificial Intelligence vs Machine Learning

- Conceito mais amplo
  - Uso de computadores para imitar funções cognitivas de seres humanos
  - Algoritmos que funcionam de maneira inteligente
- Subárea de IA
  - Foca na habilidade possuída por máquinas de receber um **conjunto de dados** e aprenderem por si próprias



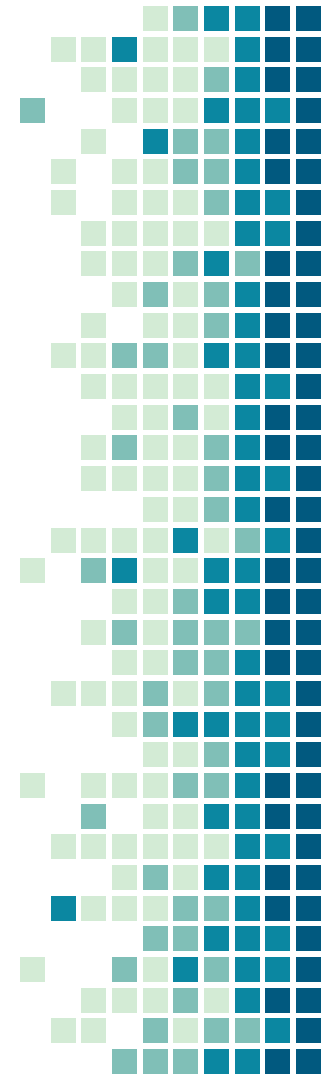
# Data Science

vs

# Data Mining

- Uma área
- Multidisciplinar
- Análise social, construção de modelos preditivos, descobrimento de fatos desconhecidos, etc
- Foco na ciência

- Um conjunto de técnicas
- Faz parte de Data Science
- Achar padrões e tendências
- Foco no processo / algoritmos



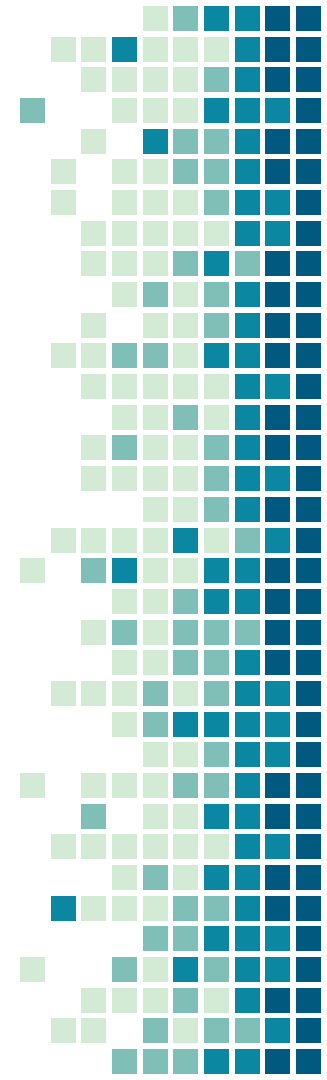
# Supervised vs Unsupervised Learning

Supervised learning trabalha em cima de dados rotulados

- **Classificação:** quando a variável de saída é uma categoria
- Regressão: quando a variável de saída é um número real
- Algoritmos: SVM, Random Forest, Linear Regression, Redes neurais

Unsupervised learning só tem os dados "crus"

- Clustering: agrupamentos de dados (k-means)
- Association: quando se quer descobrir regras que governam os dados como "pessoas que compram X também tendem a comprar Y" (Apriori algorithm)



# E quanto a redes neurais vs deep learning?

Guardem no buffer...



# 5.

## Redes Neurais Artificiais, Teoria

Primeiro vamos entender teoricamente  
como uma ANN funciona.





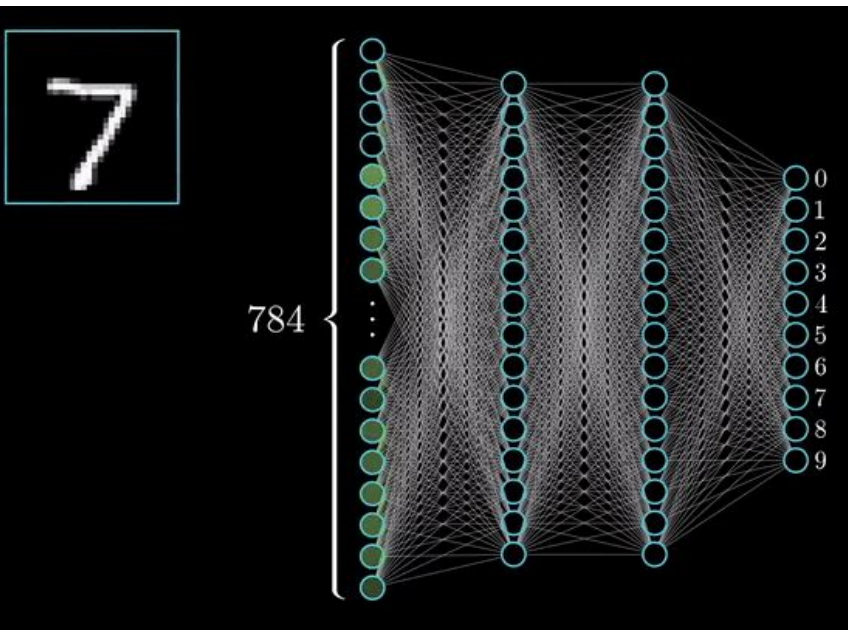


- Como seu cérebro consegue classificar cada dígito?
- Como você escreveria um algoritmo para identificar cada imagem como um dígito específico?
- Tarefas complexas!

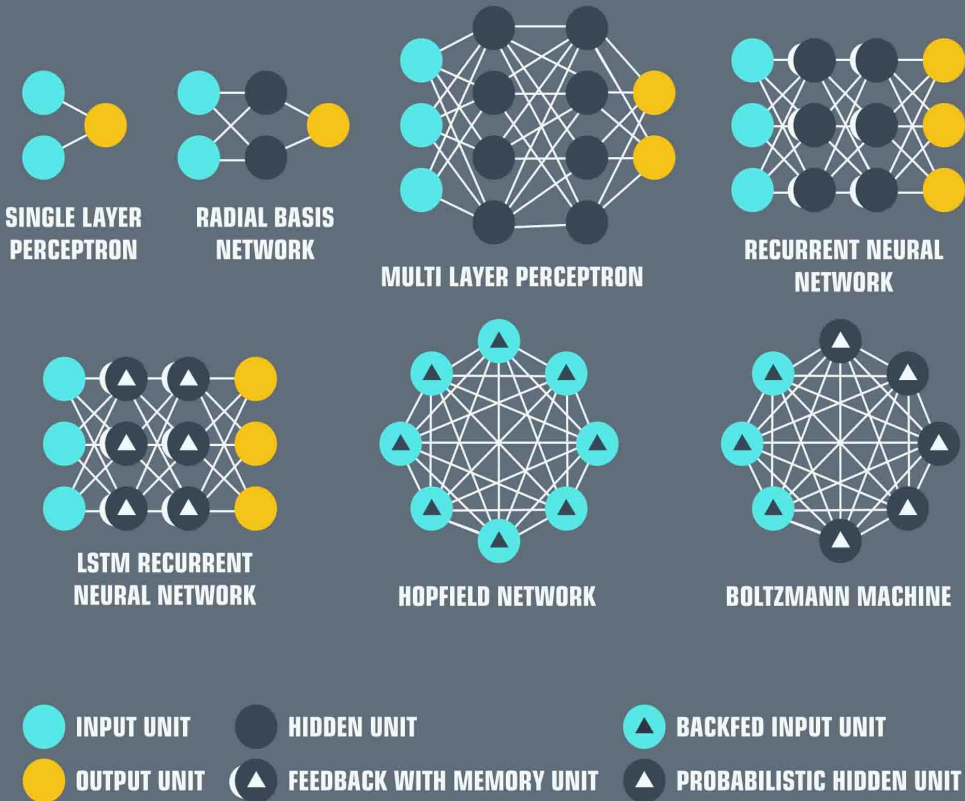


# Há diversas variações de redes neurais

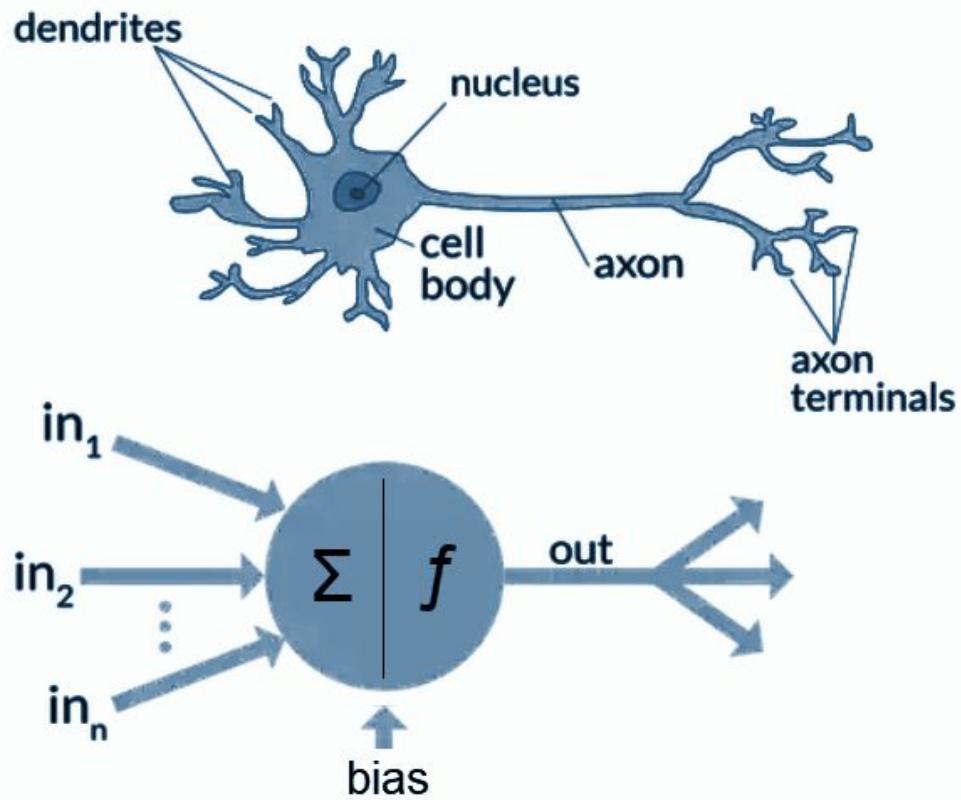
Neste curso focaremos em Multilayer Perceptron (obrigado 3b1b)



## NEURAL NETWORK ARCHITECTURE TYPES

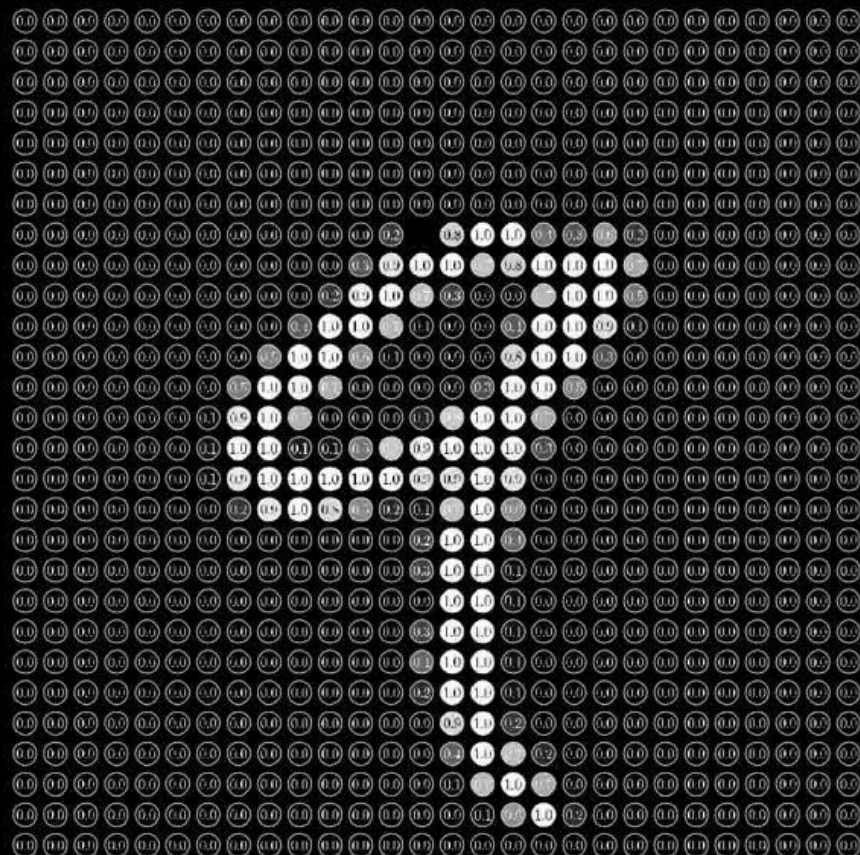


# Inspiração Biológica



28

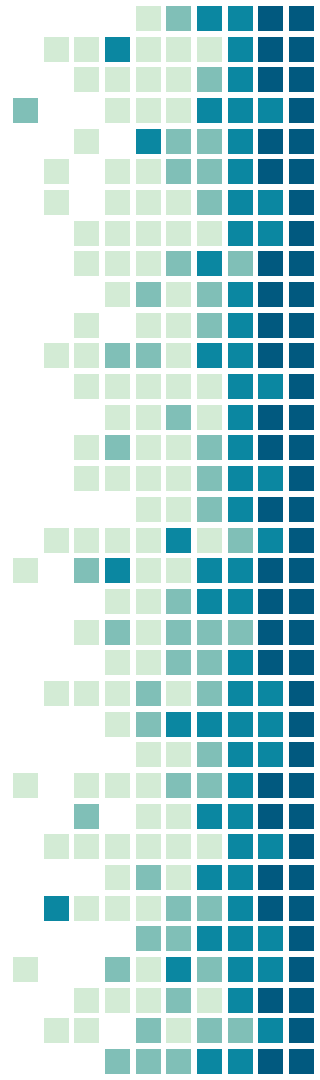
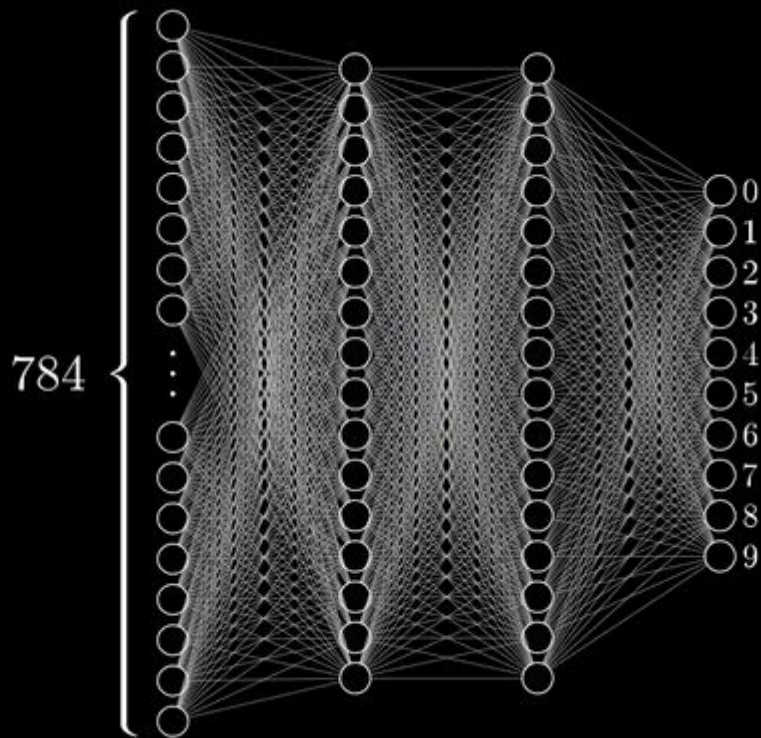
28



$$28 \times 28 = 784$$

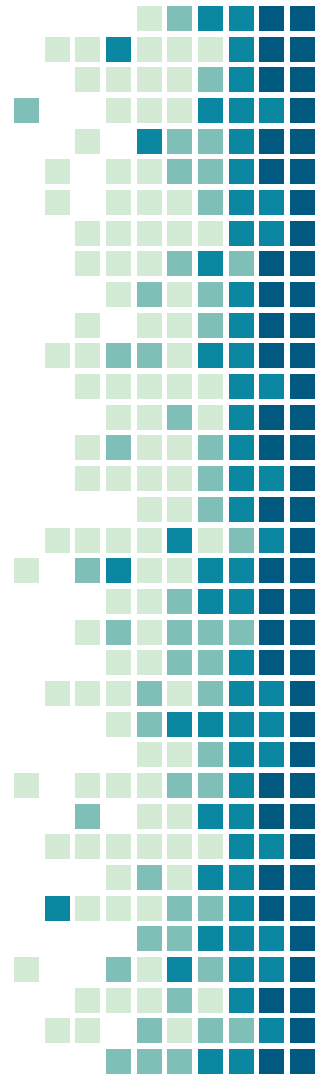
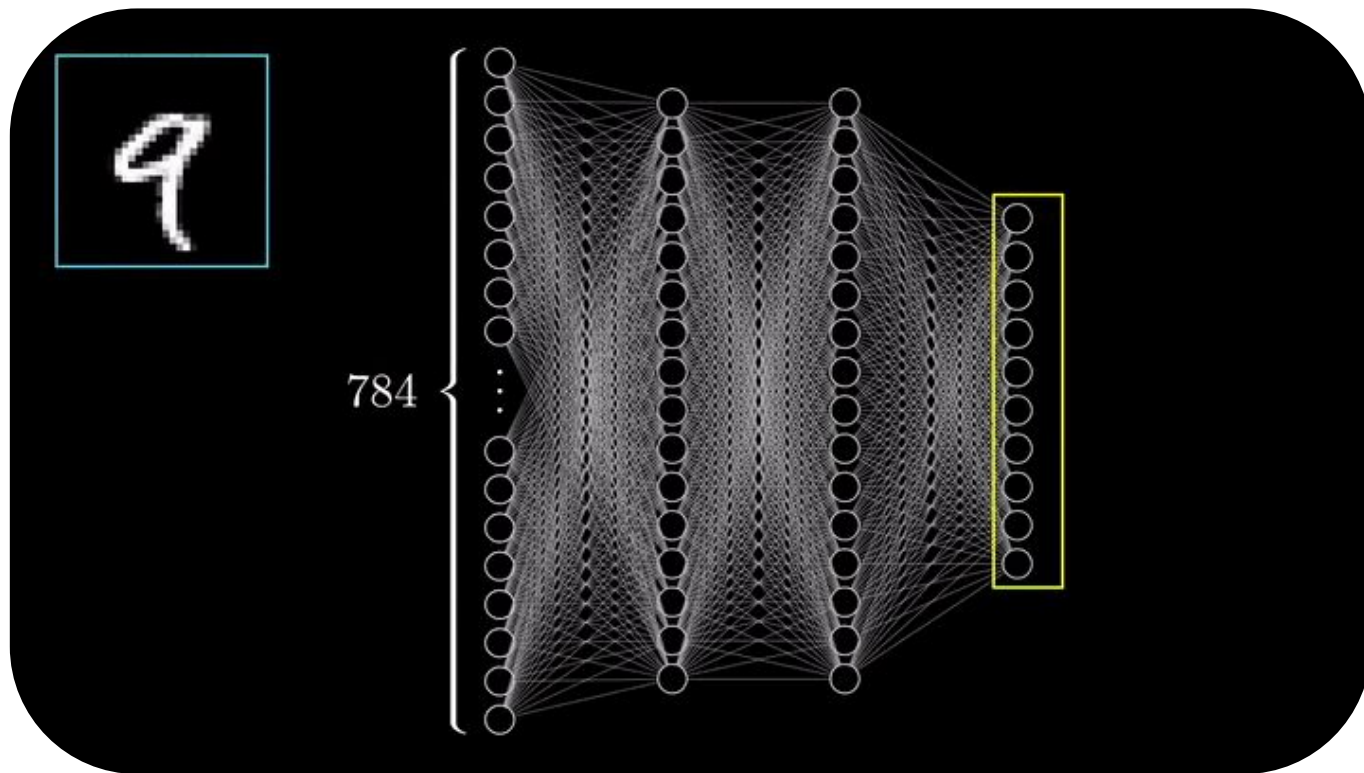
0.58

“Activation”





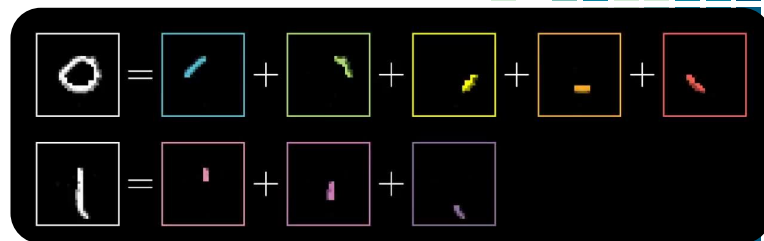
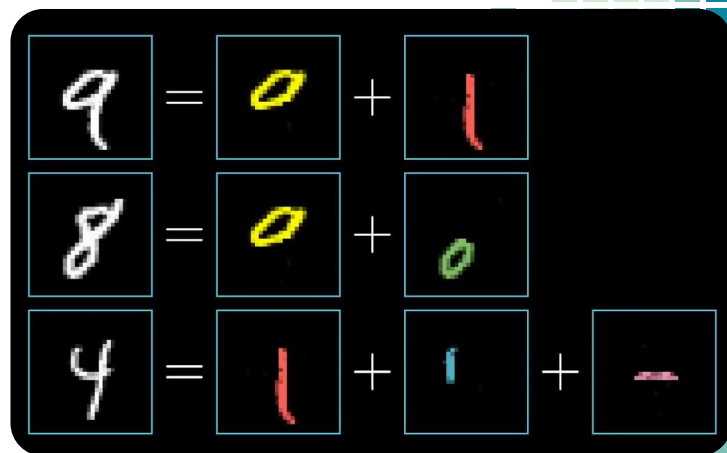
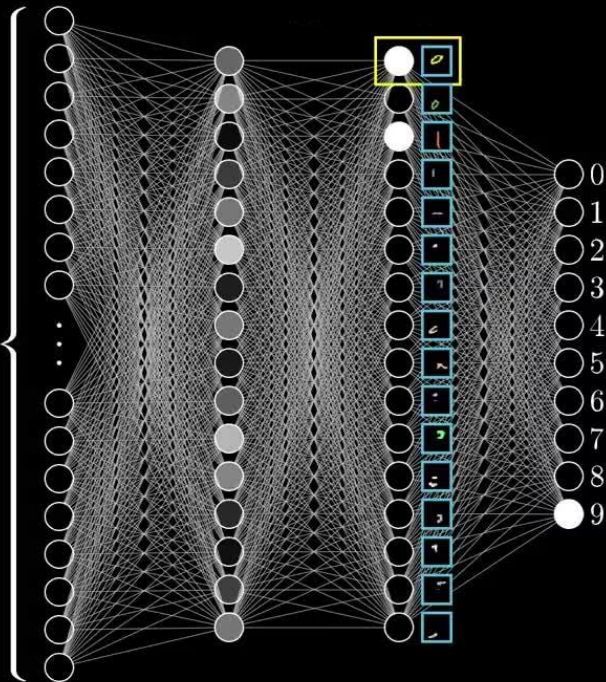
# A camada de Output



# Hidden Layers



784



# HIDDEN LAYERS

- Quebrar tarefas complexas
- Melhorar performance
- Muitos exemplos podem explorar essa característica

Reconhecimento facial, reconhecimento de fala, etc

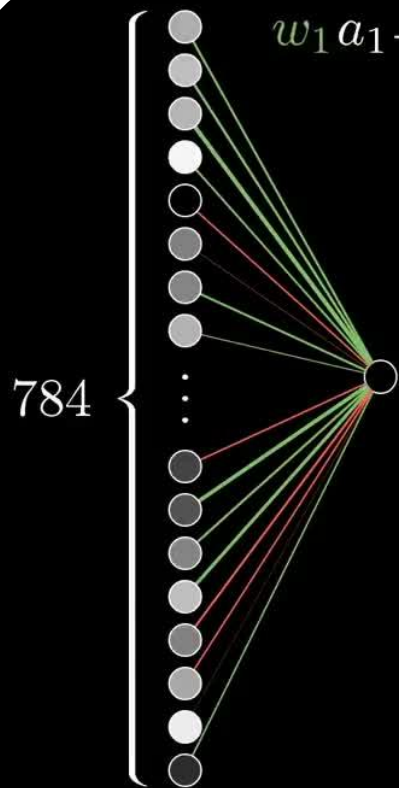




COMO FAZEMOS AS  
ATIVAC<sub>3</sub>ÕES DE UMA  
CAMADA AFETAR AS  
OUTRAS?



# Pesos!



$$w_1 a_1 + w_2 a_2 + w_3 a_3 + w_4 a_4 + \dots + w_n a_n$$

$$w_1: 2.07$$

$$w_2: 2.31$$

$$w_3: 3.64$$

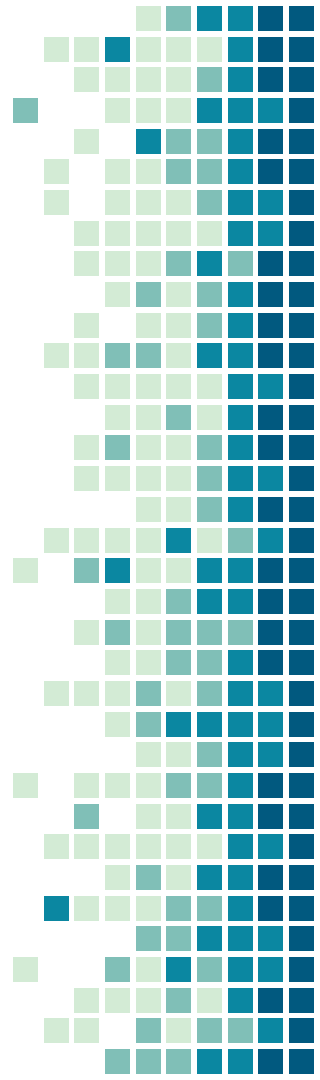
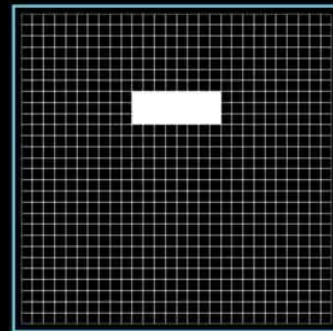
$$w_4: 1.87$$

$$w_5: -1.51$$

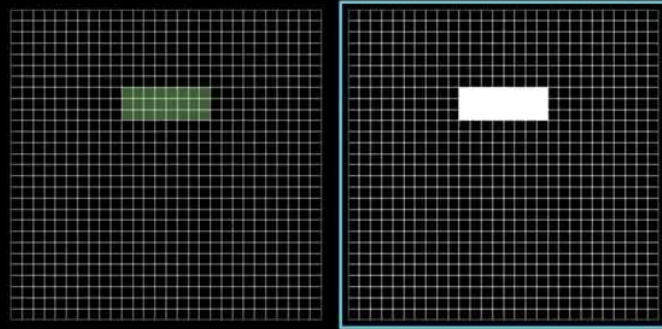
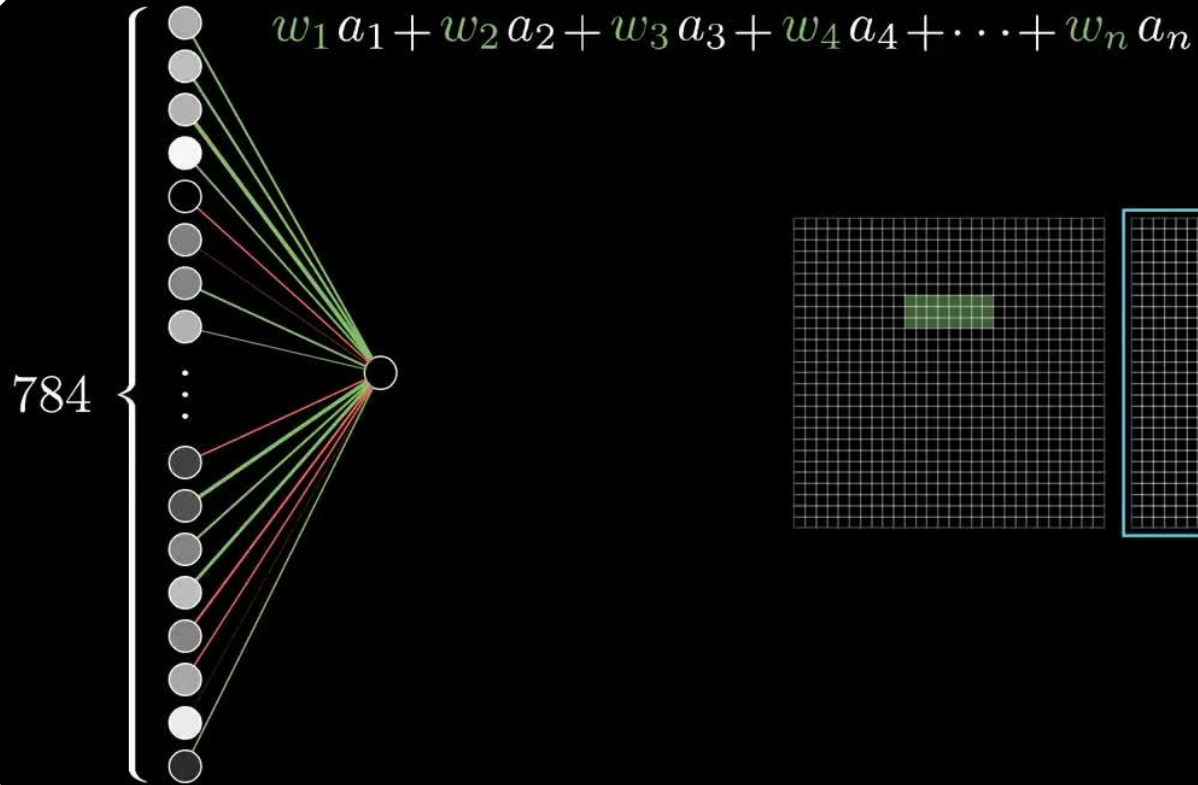
$$w_6: -0.43$$

$$w_7: 2.01$$

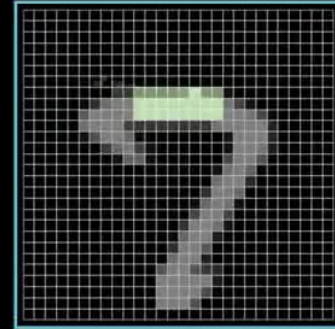
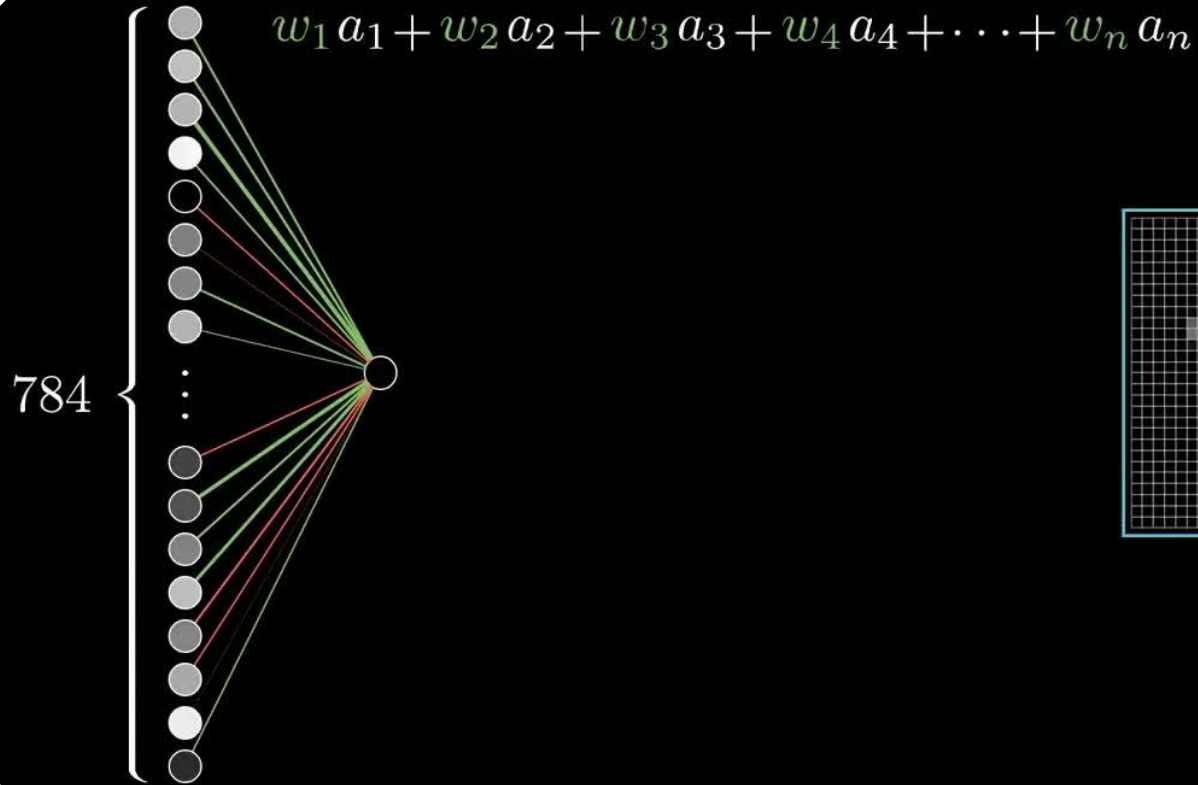
$$w_8: 1.07$$

$$\vdots$$


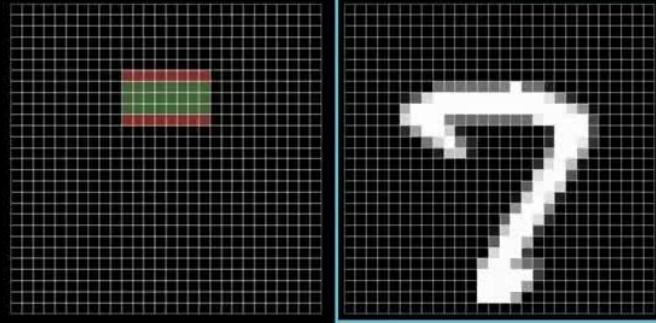
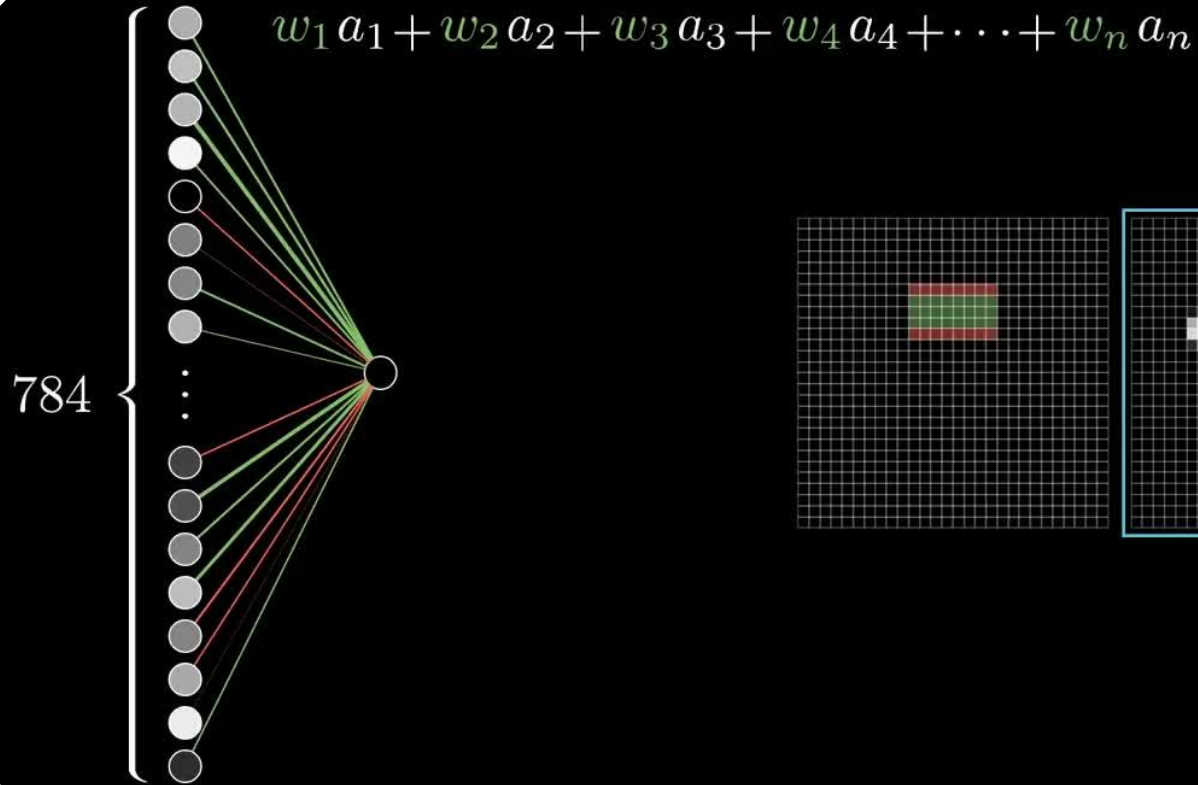
# Pesos!



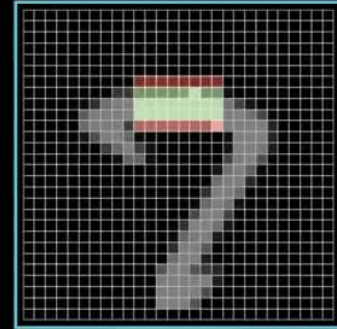
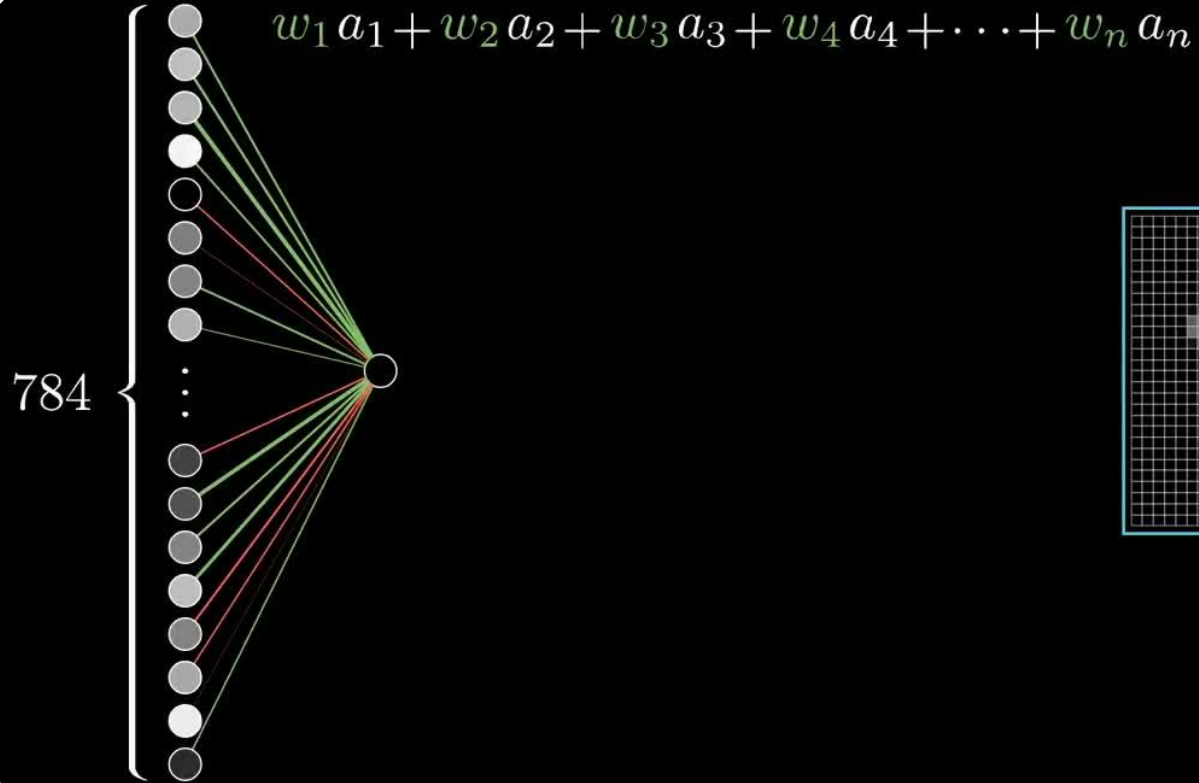
# Pesos!



# Pesos!

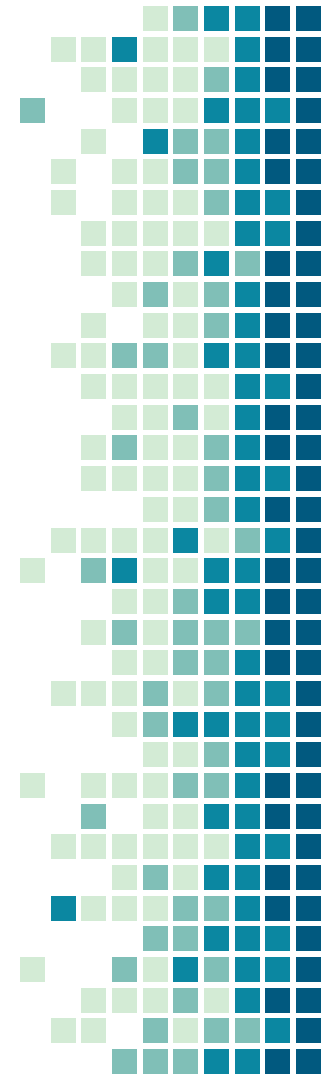


# Pesos!



# Problema: valor de ativação muito alto

$$w_1a_1 + w_2a_2 + w_3a_3 + w_4a_4 + \cdots + w_na_n$$



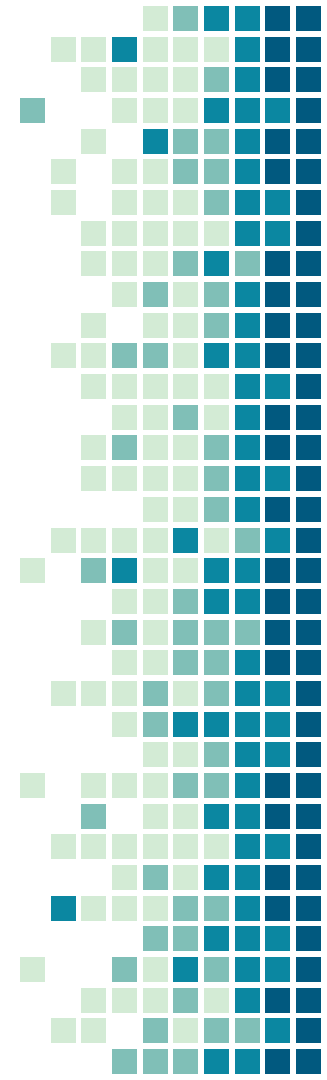
# Solução: espremer valores num intervalo

$$w_1a_1 + w_2a_2 + w_3a_3 + w_4a_4 + \cdots + w_na_n$$

-4 -3 -2 -1 0 1 2 3 4

-4 -3 -2 -1 0 1 2 3 4

Activations should be in this range

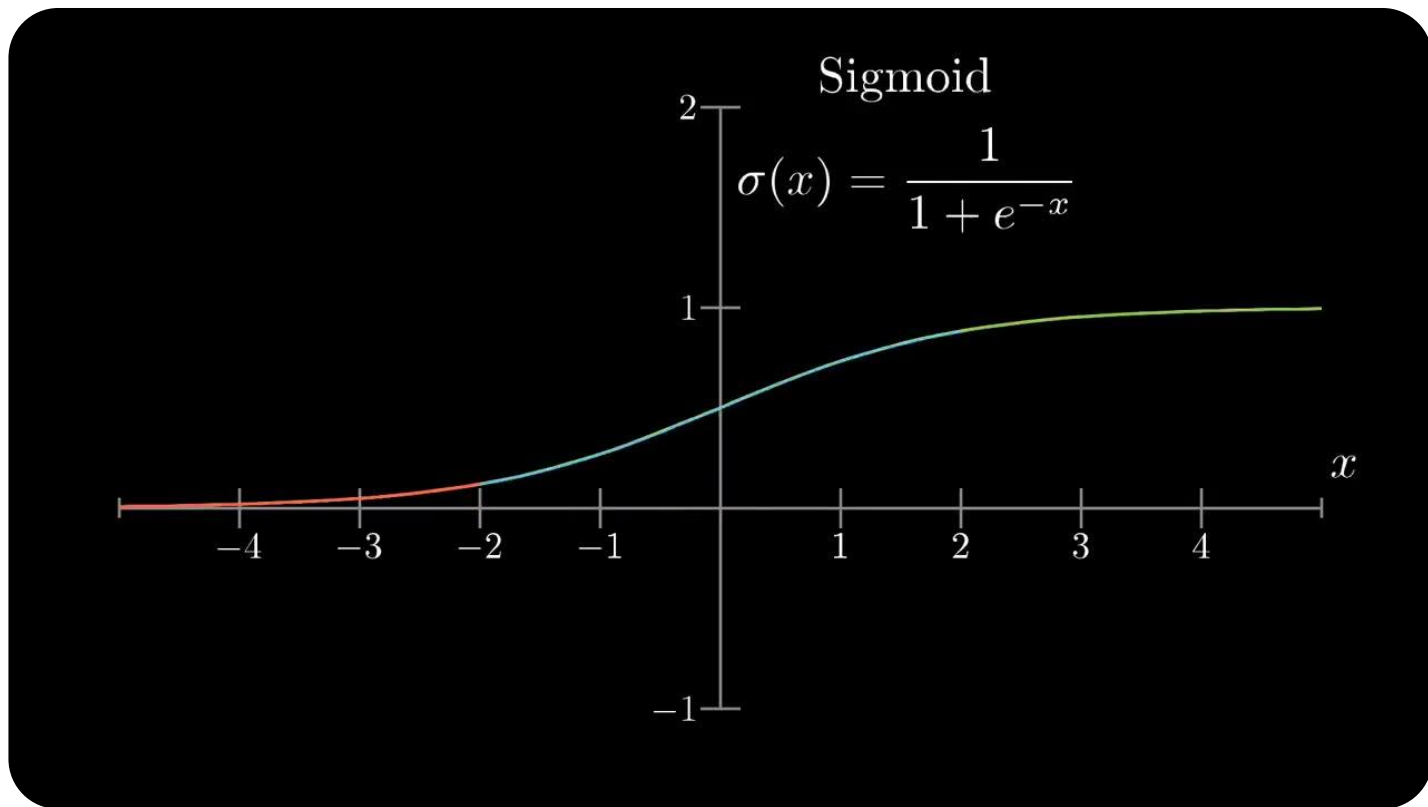




COMO FAZER COM  
QUE AS ATIVAÇÕES  
ESTEJAM  
DELIMITADAS EM UM  
INTERVALO  
ESPECÍFICO?



# Função de Ativação



# Bias

Sigmoid

How positive is this?

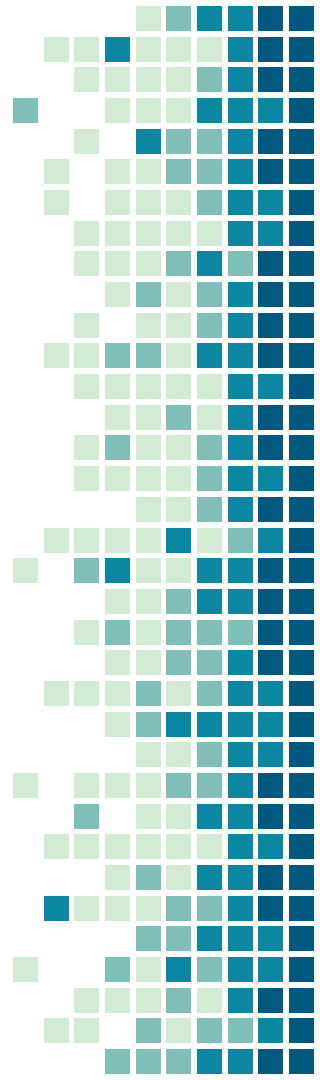
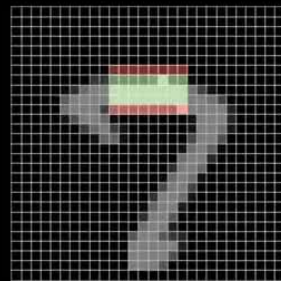
$$\sigma(w_1 a_1 + w_2 a_2 + w_3 a_3 + \cdots + w_n a_n \boxed{-10})$$

“bias”

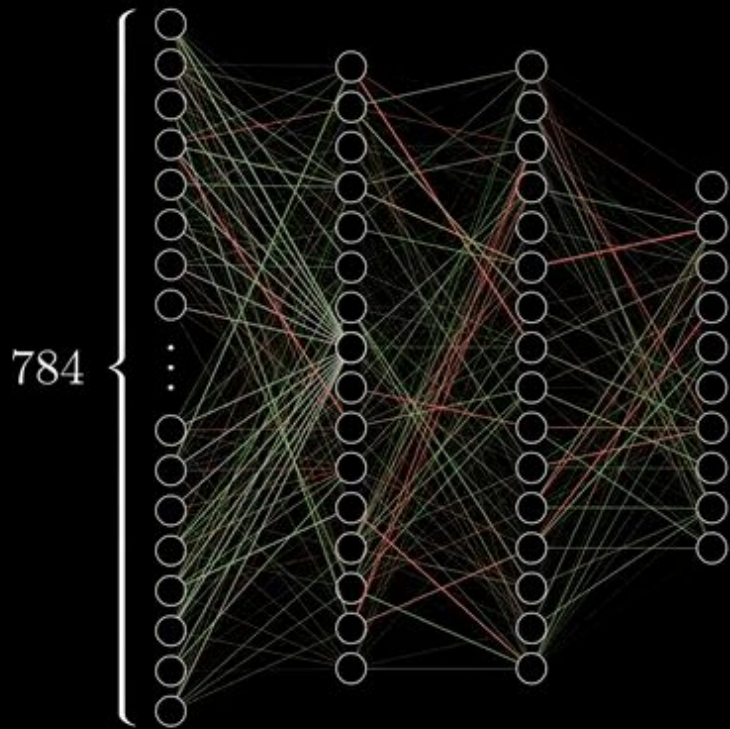
784

:

Only activate meaningfully  
when **weighted sum**  $> 10$



E isso para todos os neurônios...



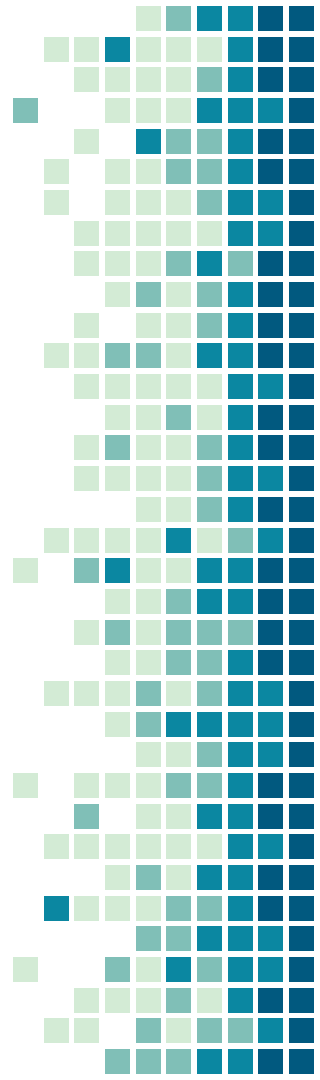
$$784 \times 16 + 16 \times 16 + 16 \times 10$$

weights

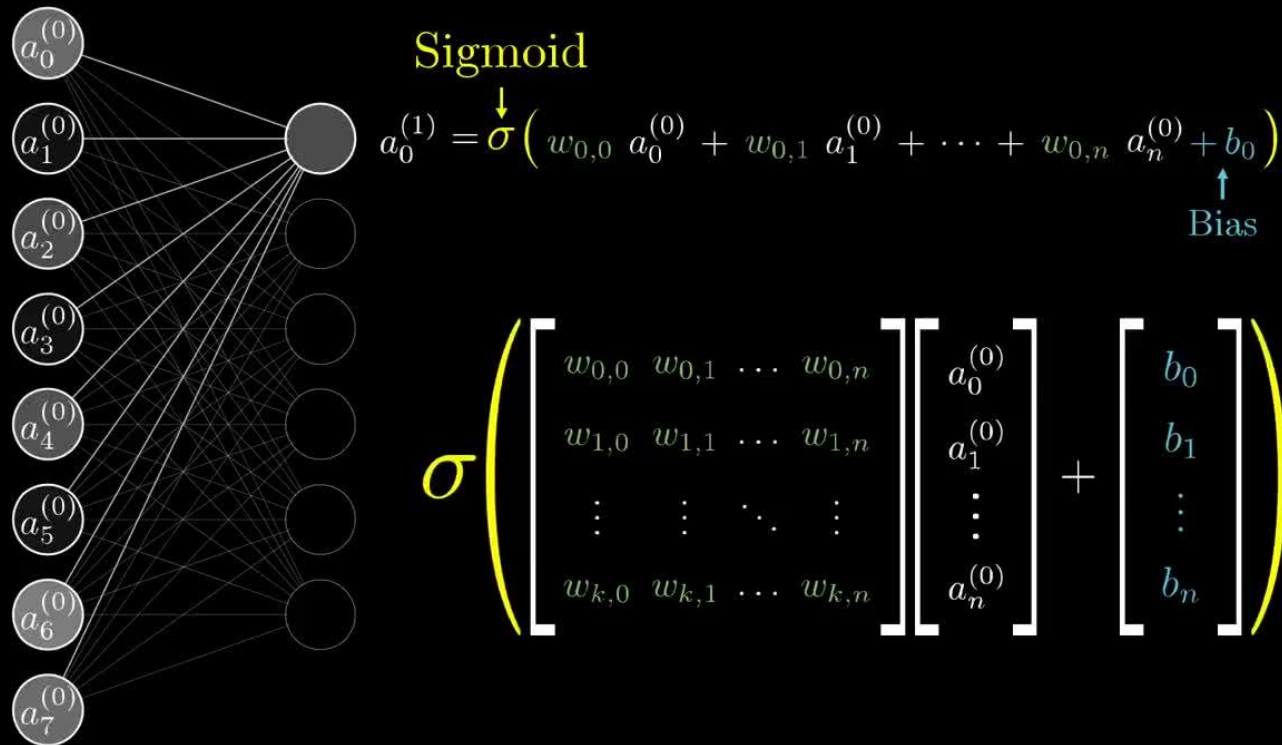
$$16 + 16 + 10$$

biases

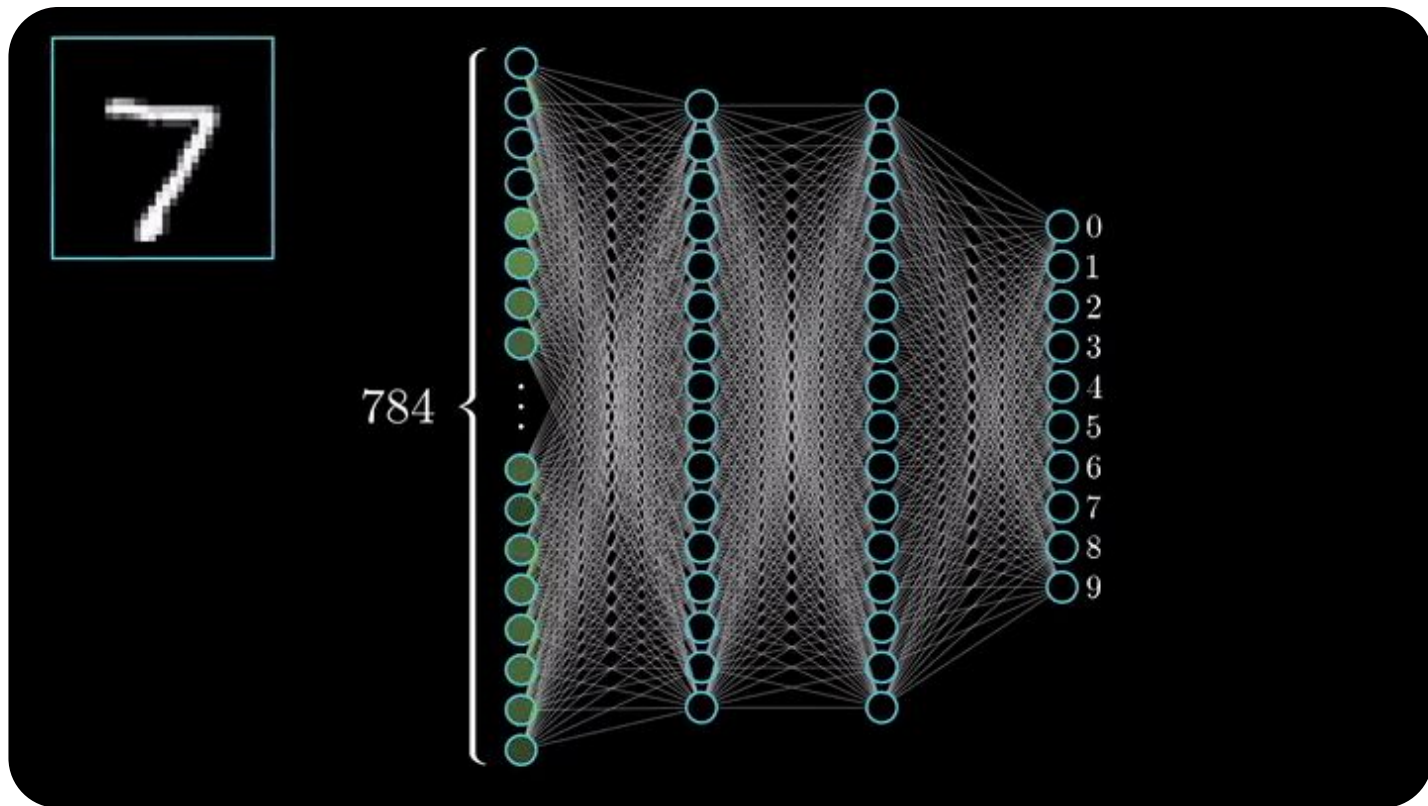
$$13,002$$



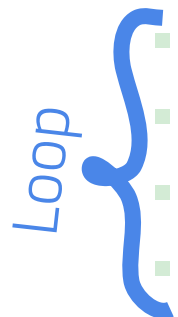
# Representação compacta formal



# Feedforward

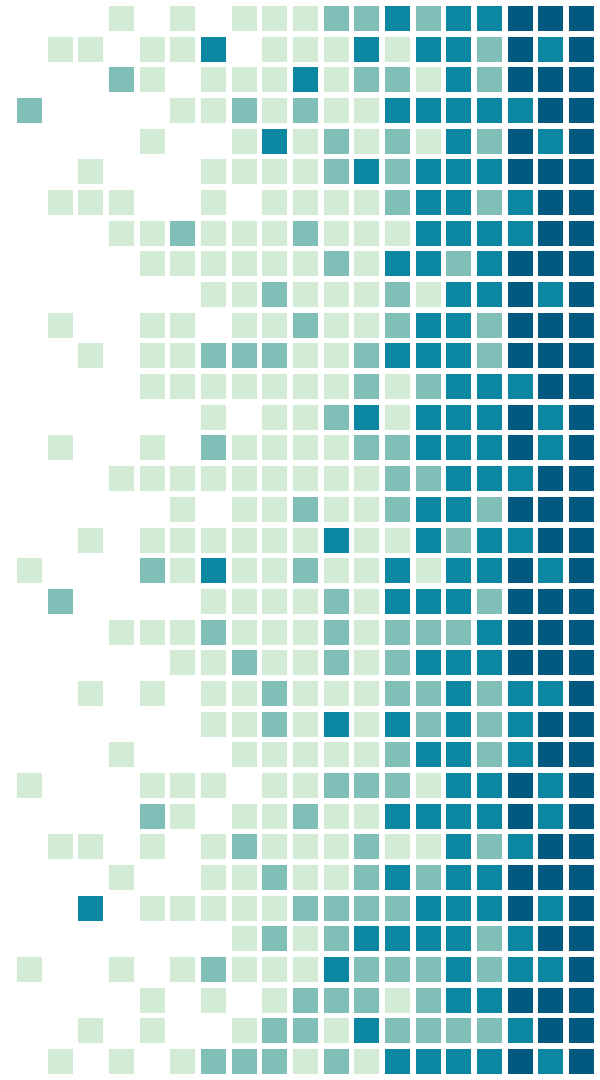


# O PROCESSO DE APRENDIZAGEM

- 
- 1. Iniciamos a rede com pesos e bias aleatórios
  - 2. Alimentamos com uma amostra
  - 3. Calculamos o erro do output
  - 4. Atualizamos os pesos baseado nesse erro
  - 5. Retornamos ao passo 2.



COMO CALCULAR O  
ERRO ENTRE A SAÍDA  
DA REDE NEURAL E A  
SAÍDA IDEAL?





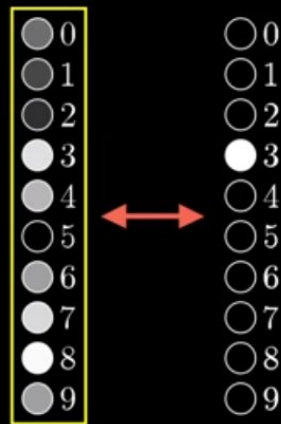
# Cost Functions!

Cost of 3

3.37

$$\left\{ \begin{array}{l} 0.1863 \leftarrow (0.43 - 0.00)^2 + \\ 0.0809 \leftarrow (0.28 - 0.00)^2 + \\ 0.0357 \leftarrow (0.19 - 0.00)^2 + \\ 0.0138 \leftarrow (0.88 - 1.00)^2 + \\ 0.5242 \leftarrow (0.72 - 0.00)^2 + \\ 0.0001 \leftarrow (0.01 - 0.00)^2 + \\ 0.4079 \leftarrow (0.64 - 0.00)^2 + \\ 0.7388 \leftarrow (0.86 - 0.00)^2 + \\ 0.9817 \leftarrow (0.99 - 0.00)^2 + \\ 0.3998 \leftarrow (0.63 - 0.00)^2 \end{array} \right.$$

What's the “cost”  
of this difference?



Utter trash

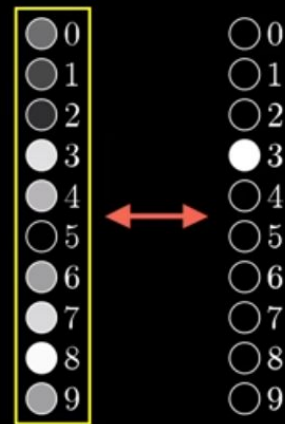
# Alto quando ela ainda não foi treinada

Cost of 3

3.37

$$\left\{ \begin{array}{l} 0.1863 \leftarrow (0.43 - 0.00)^2 + \\ 0.0809 \leftarrow (0.28 - 0.00)^2 + \\ 0.0357 \leftarrow (0.19 - 0.00)^2 + \\ 0.0138 \leftarrow (0.88 - 1.00)^2 + \\ 0.5242 \leftarrow (0.72 - 0.00)^2 + \\ 0.0001 \leftarrow (0.01 - 0.00)^2 + \\ 0.4079 \leftarrow (0.64 - 0.00)^2 + \\ 0.7388 \leftarrow (0.86 - 0.00)^2 + \\ 0.9817 \leftarrow (0.99 - 0.00)^2 + \\ 0.3998 \leftarrow (0.63 - 0.00)^2 \end{array} \right.$$

What's the “cost”  
of this difference?



Utter trash

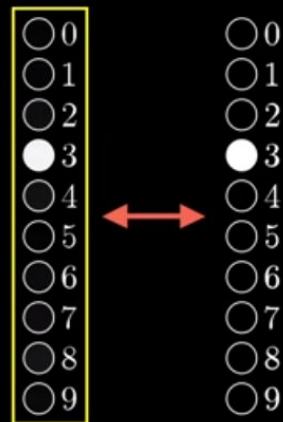
# Baixo quando ela se aproxima do correto

Cost of 3

0.03

{	0.0006	←	$(0.02 - 0.00)^2 +$
	0.0007	←	$(0.03 - 0.00)^2 +$
	0.0039	←	$(0.06 - 0.00)^2 +$
	0.0009	←	$(0.97 - 1.00)^2 +$
	0.0055	←	$(0.07 - 0.00)^2 +$
	0.0004	←	$(0.02 - 0.00)^2 +$
	0.0022	←	$(0.05 - 0.00)^2 +$
	0.0033	←	$(0.06 - 0.00)^2 +$
	0.0072	←	$(0.08 - 0.00)^2 +$
	0.0018	←	$(0.04 - 0.00)^2$

What's the “cost” of this difference?

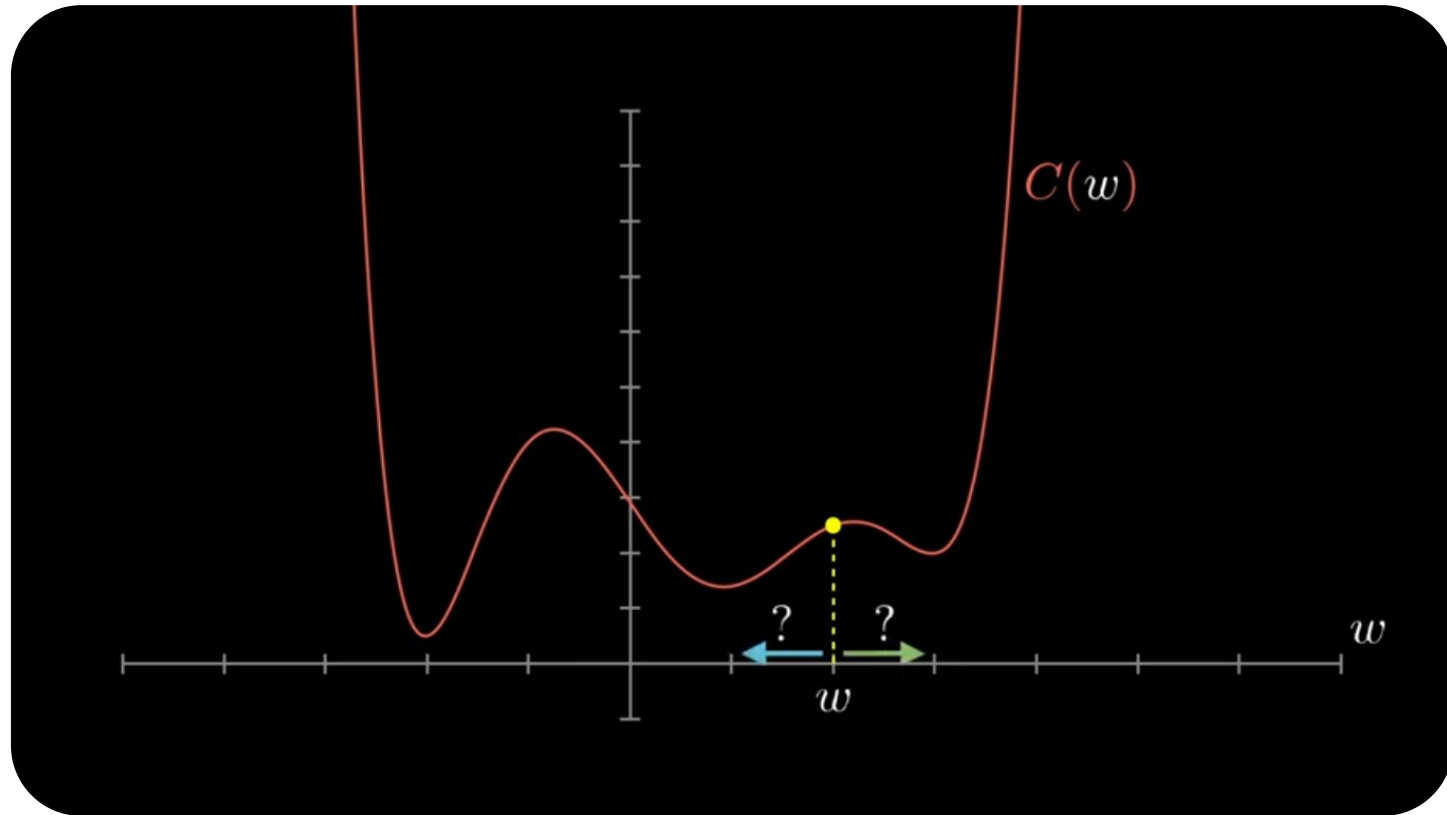


Utter trash

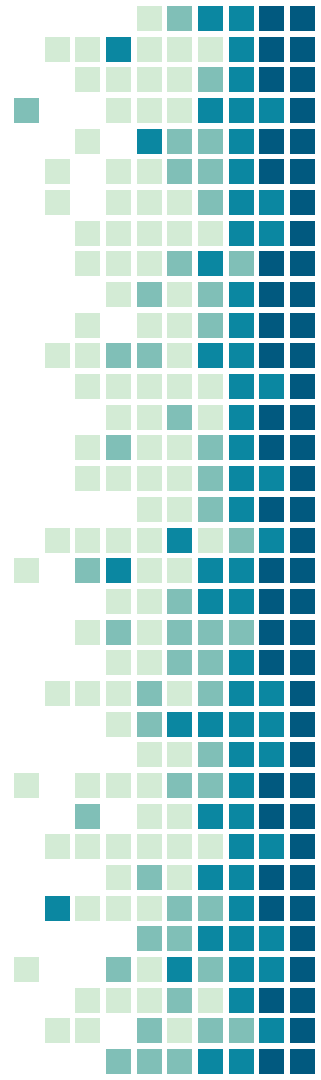
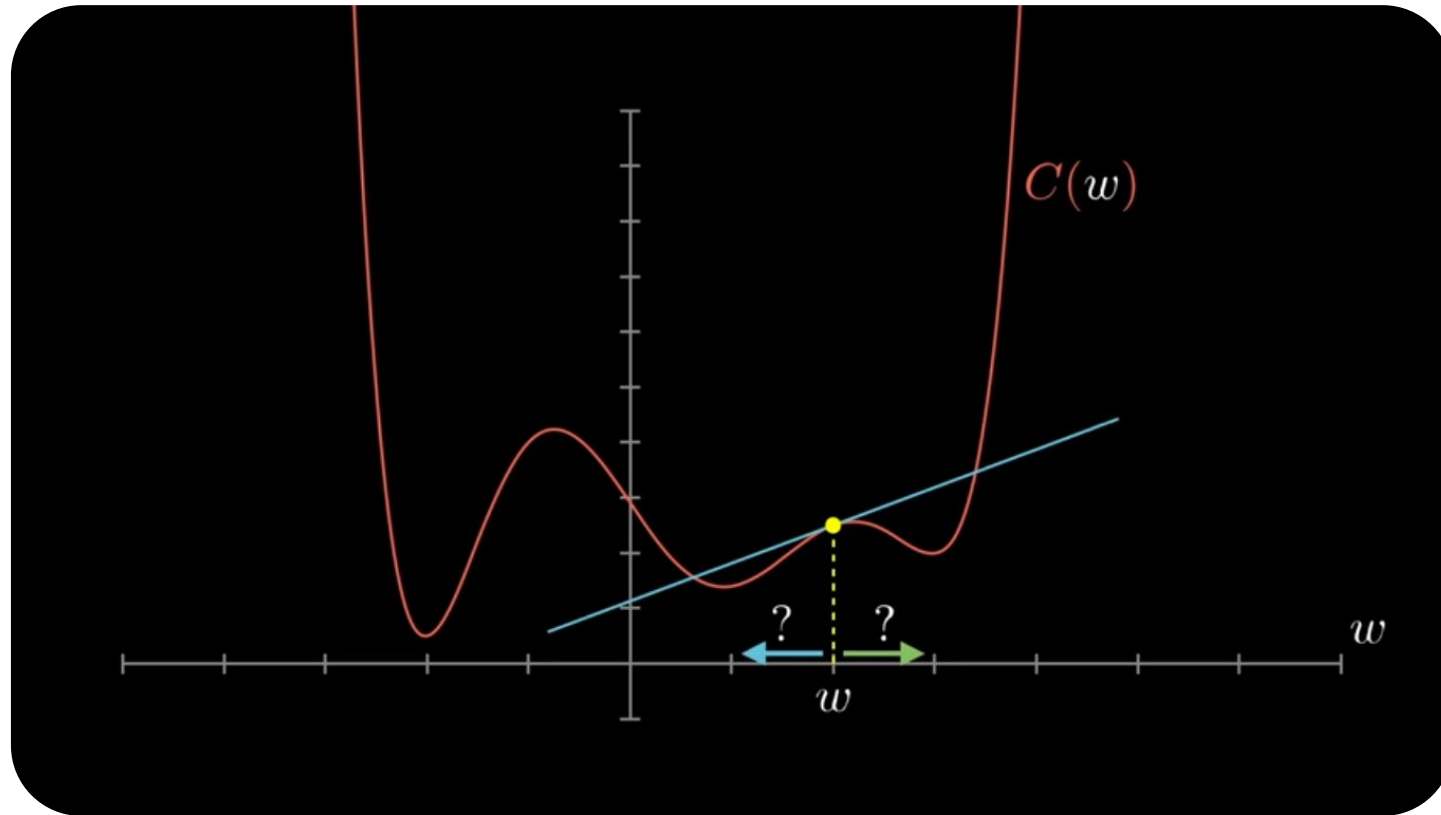
CALCULADO O CUSTO,  
COMO ATUALIZAMOS  
OS VALORES DOS  
PESOS?



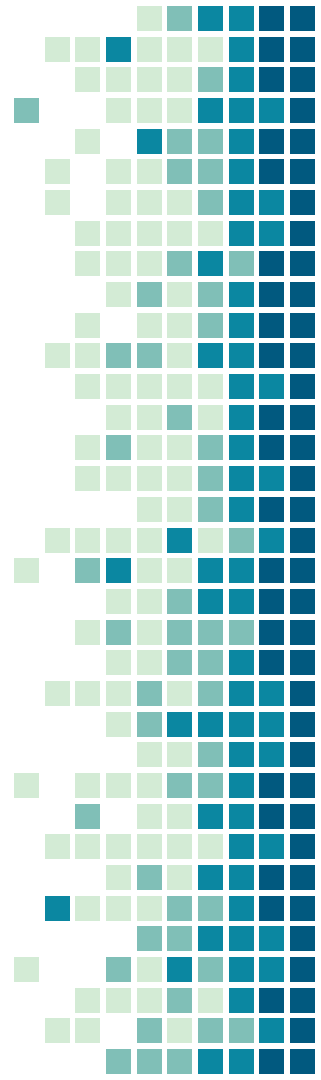
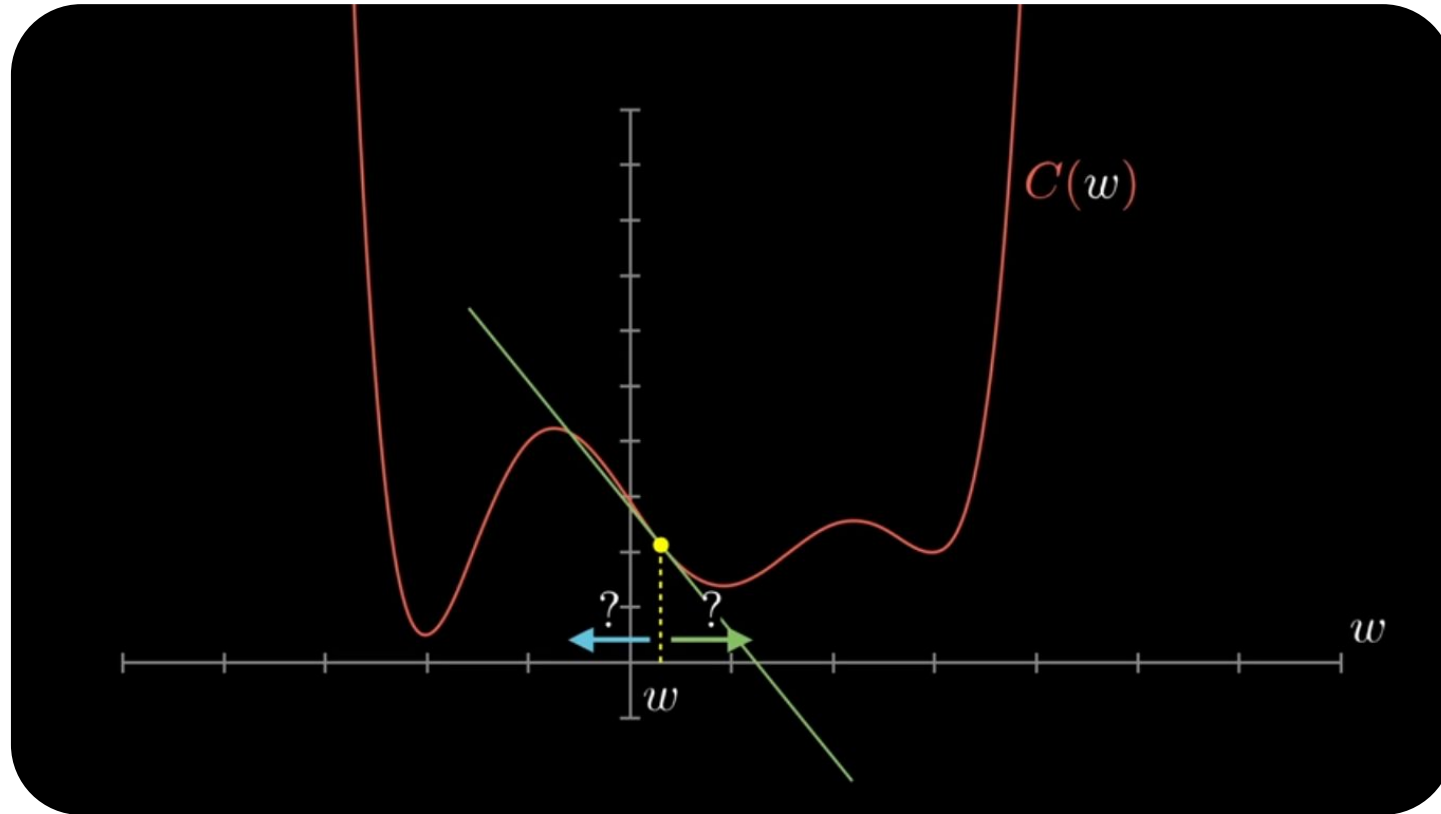
# Mínimos da Função



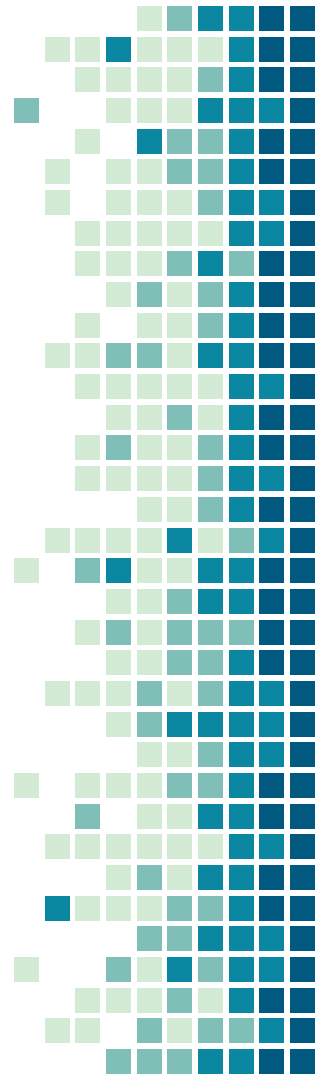
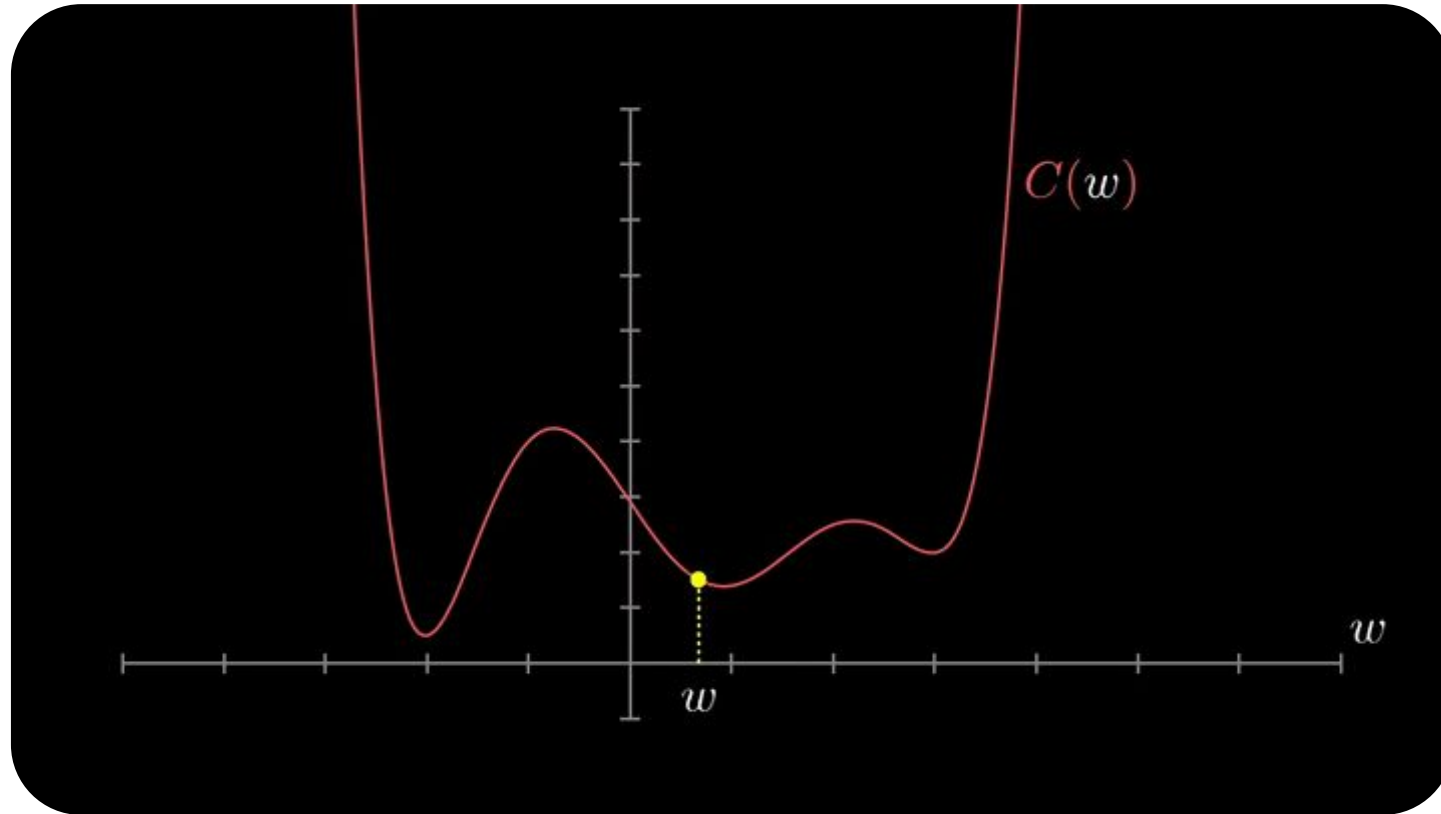
# Mínimos da Função



# Mínimos da Função

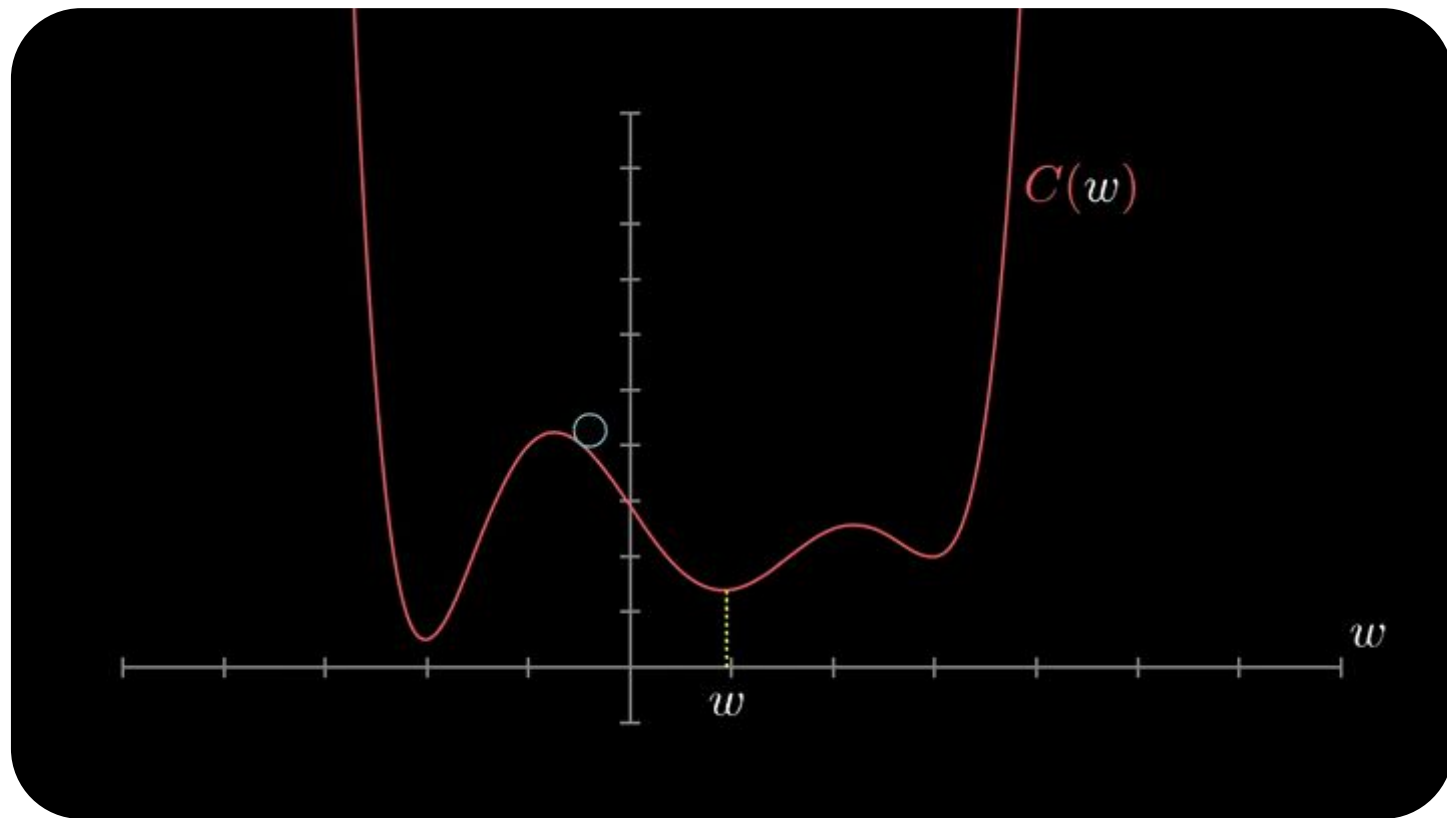


# Mínimos da Função

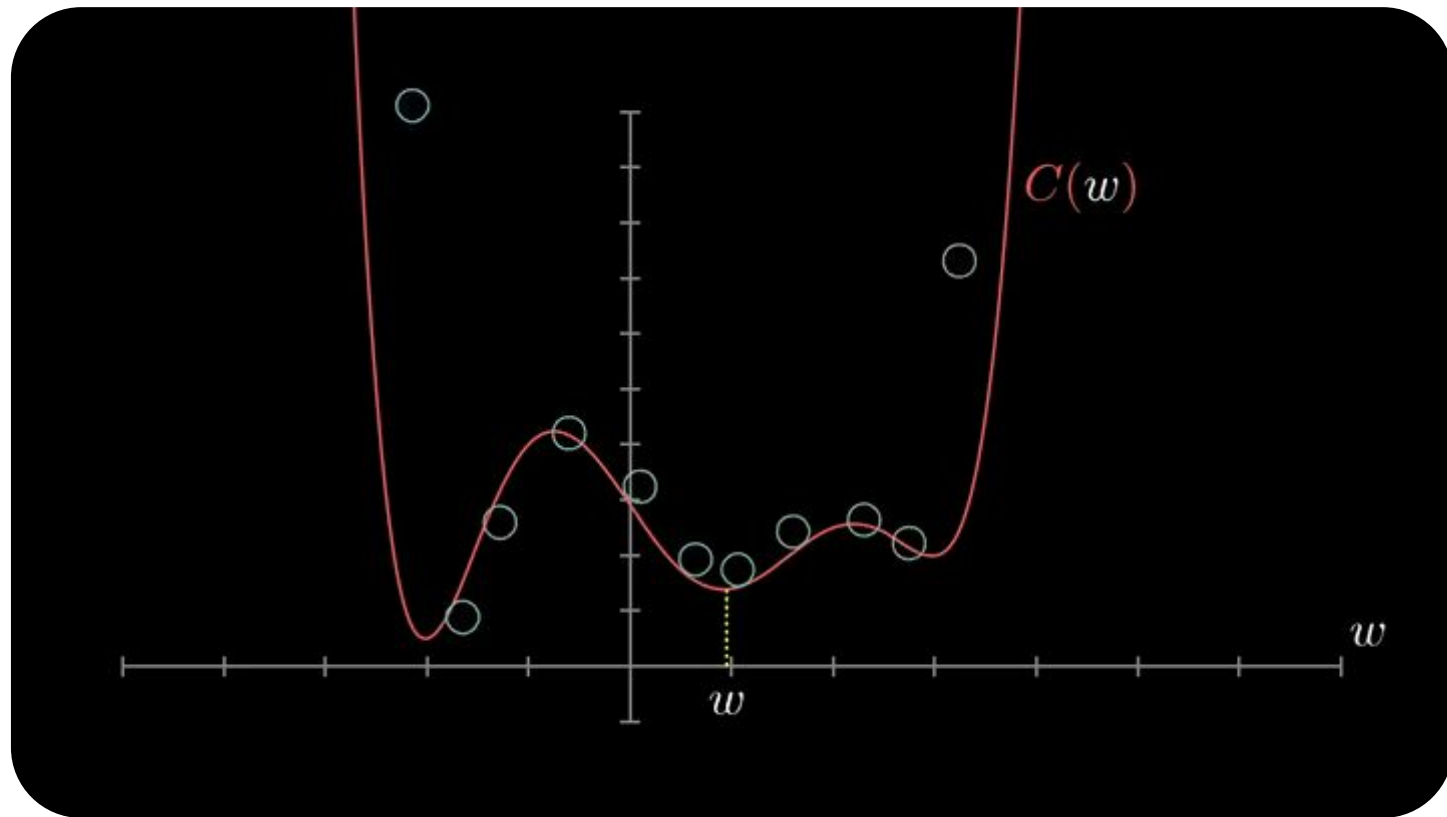




“Bola descendo um morro”



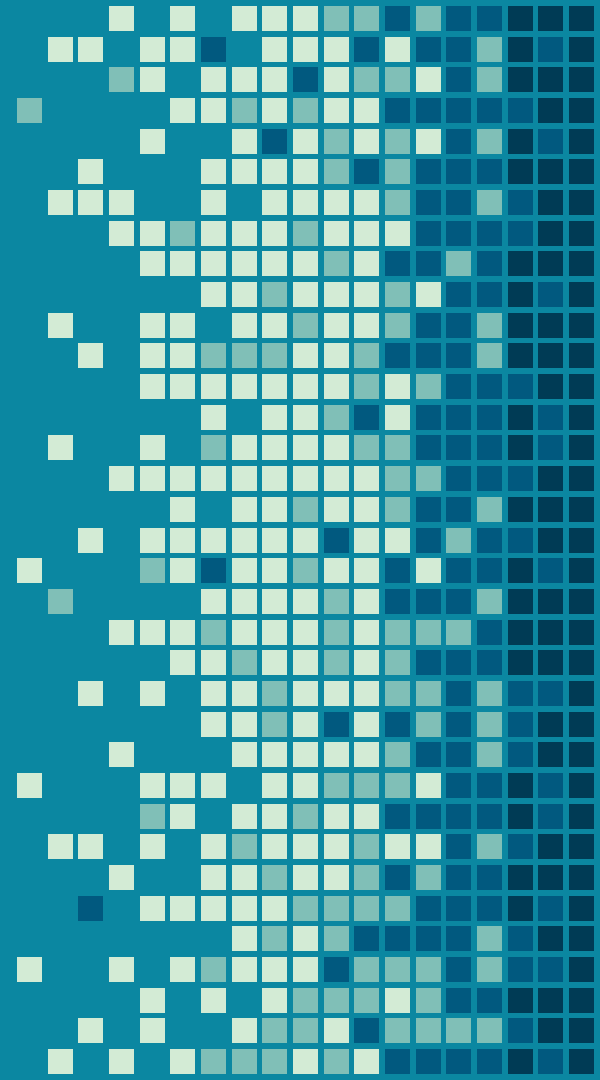
# Vários mínimos locais



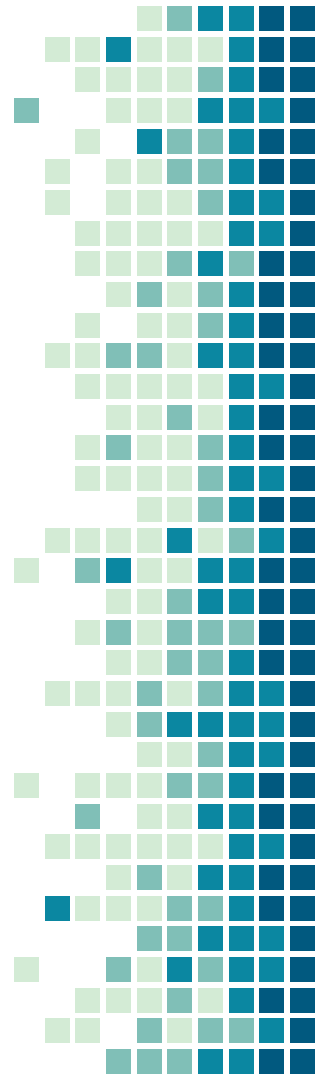
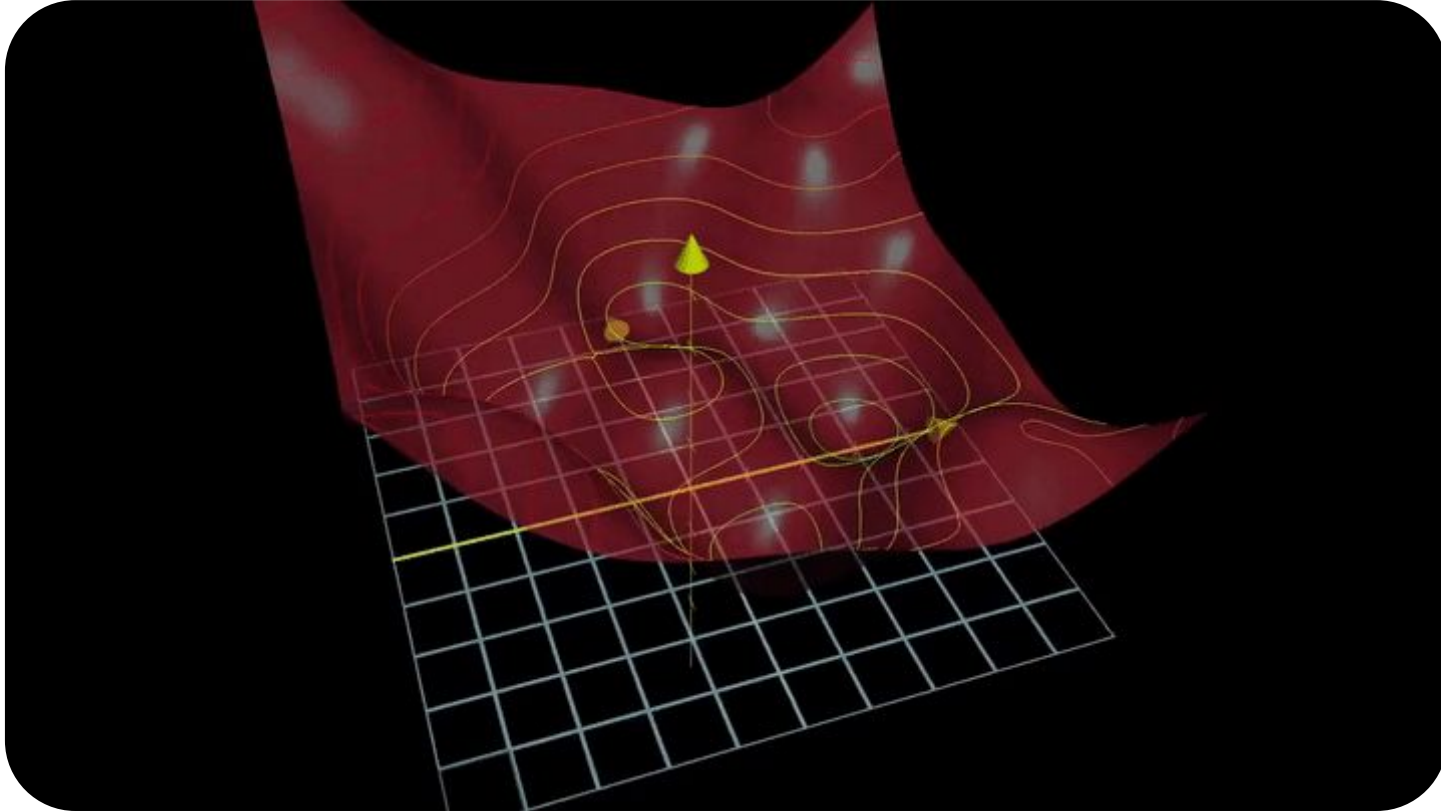
“

*Você pode acabar em diversos vales distintos dependendo do input randômico inicial.*

*Não há nenhuma garantia de que o mínimo local encontrado seja o menor valor possível para a função de custo.*



# O algoritmo para minimizar a função



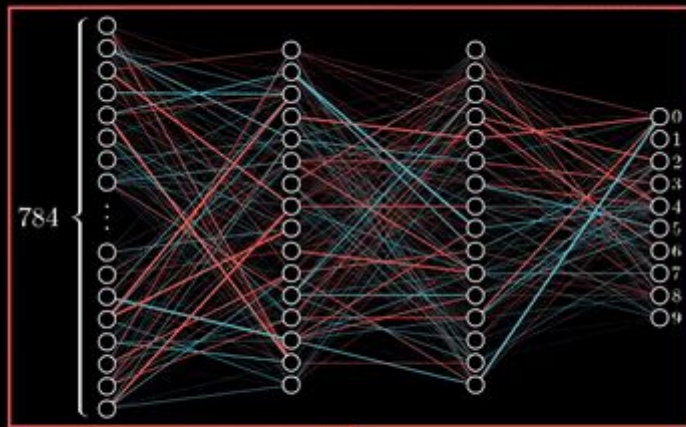
# GRADIENT DESCENT

- “Método do gradiente” / “método do máximo declive”
- Procura mínimos locais na função
- Na prática nos diz como alterar os pesos e bias das conexões para diminuir o custo eficientemente



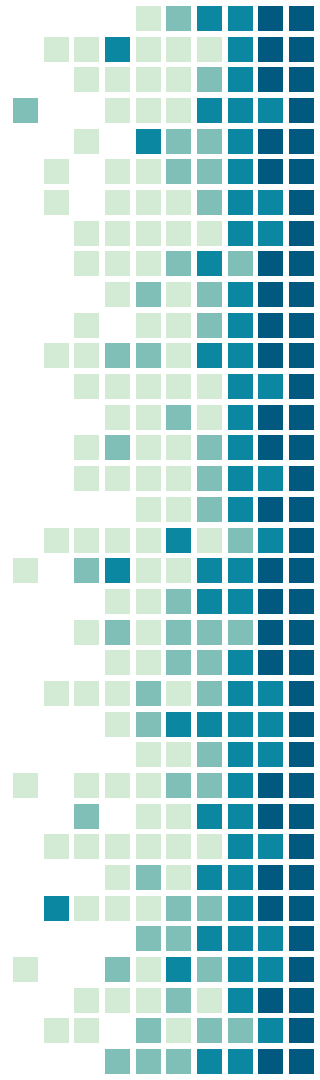
# Backtracking

$$-\nabla C(\underbrace{\dots}_{\text{All weights and biases}}) = \begin{bmatrix} 0.20 \\ 0.83 \\ -0.84 \\ \vdots \\ 0.04 \\ 1.57 \\ 1.59 \end{bmatrix}$$

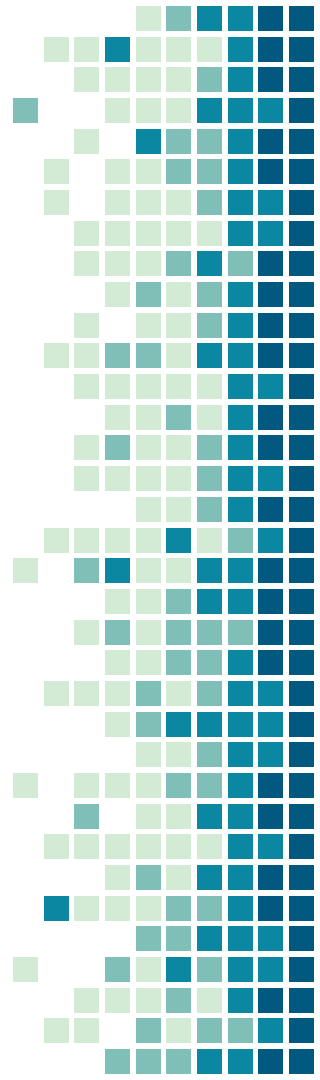


↓

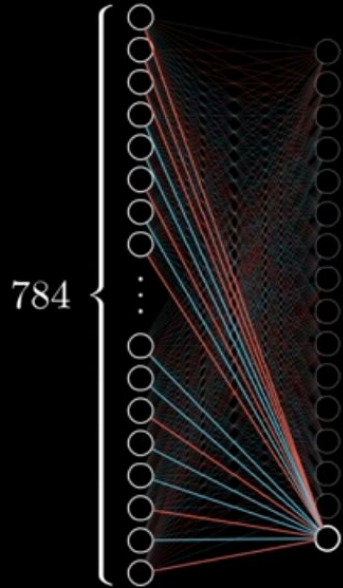
$$C(w_0, w_1, \dots, w_{13,001}) = 3.4$$



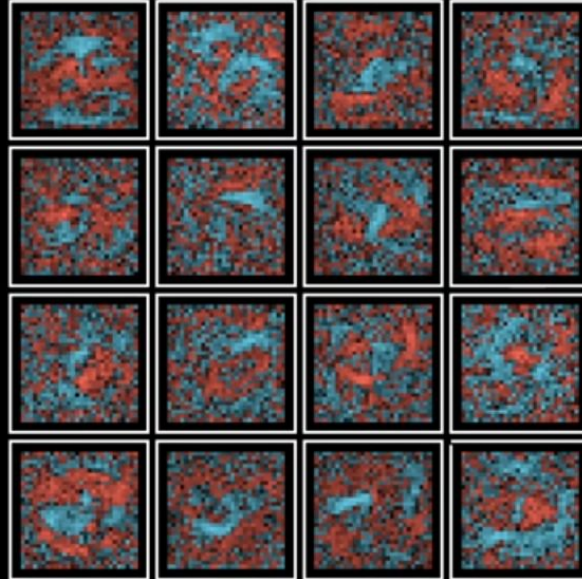
# Hidden layers na prática...



# Hidden layers na prática... Alquimia.



What second layer  
neurons look for





# 6. Keras

Redes neurais artificiais na prática



# O que é Keras?

- Biblioteca **open source** licenciada pelo MIT
- Cross-Platform: TensorFlow (que utilizaremos), Microsoft Cognitive Toolkit, Theano ou PlaidML
- Construção de redes neurais
- Escrita em Python



# Por que utilizar Keras?

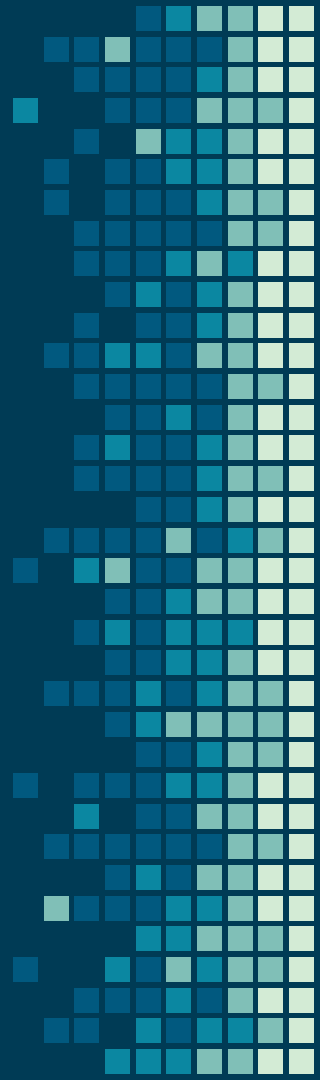
- Setup de modelos rápido
- User-friendly
- Experimentação
- Backend otimizado





# HANDS ON!

Bora criar nossa primeira rede neural artificial :))



# Inicialização

```
[3] import tensorflow as tf  
    model = tf.keras.models.Sequential()
```

# Adicionando camadas

```
[5] import tensorflow as tf

model = tf.keras.models.Sequential()

model.add(tf.keras.layers.Dense(16, activation=tf.nn.sigmoid,
                                input_dim=784))
model.add(tf.keras.layers.Dense(16, activation=tf.nn.sigmoid))
model.add(tf.keras.layers.Dense(10, activation=tf.nn.sigmoid))
```

# .Flatten()

```
[14] import tensorflow as tf

model = tf.keras.models.Sequential()

model.add(tf.keras.layers.Flatten())

model.add(tf.keras.layers.Dense(16, activation=tf.nn.sigmoid,
                                input_dim=784))
model.add(tf.keras.layers.Dense(16, activation=tf.nn.sigmoid))
model.add(tf.keras.layers.Dense(10, activation=tf.nn.sigmoid))
```

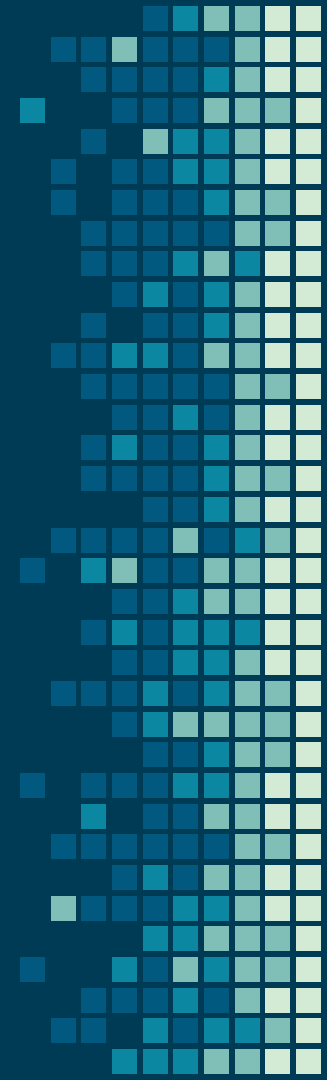
# Optimizer e loss

```
opt = tf.keras.optimizers.SGD(lr=0.1)
model.compile(loss='mean_squared_error',
              metrics=['accuracy'], optimizer=opt)
```



# Treino

```
[14] model.fit(x_train, y_train, epochs=5, batch_size=5)
```



# TL;DR

```
[ ] model = tf.keras.models.Sequential()
    model.add(tf.keras.layers.Flatten())
    model.add(tf.keras.layers.Dense(16, activation=tf.nn.sigmoid,
                                     input_dim=784))
    model.add(tf.keras.layers.Dense(16, activation=tf.nn.sigmoid))
    model.add(tf.keras.layers.Dense(10, activation=tf.nn.sigmoid))

    opt = tf.keras.optimizers.SGD(lr=0.1)
    model.compile(loss='mean_squared_error',
                  metrics=['accuracy'], optimizer=opt)

    model.fit(x_train, y_train, epochs=5, batch_size=5)
```