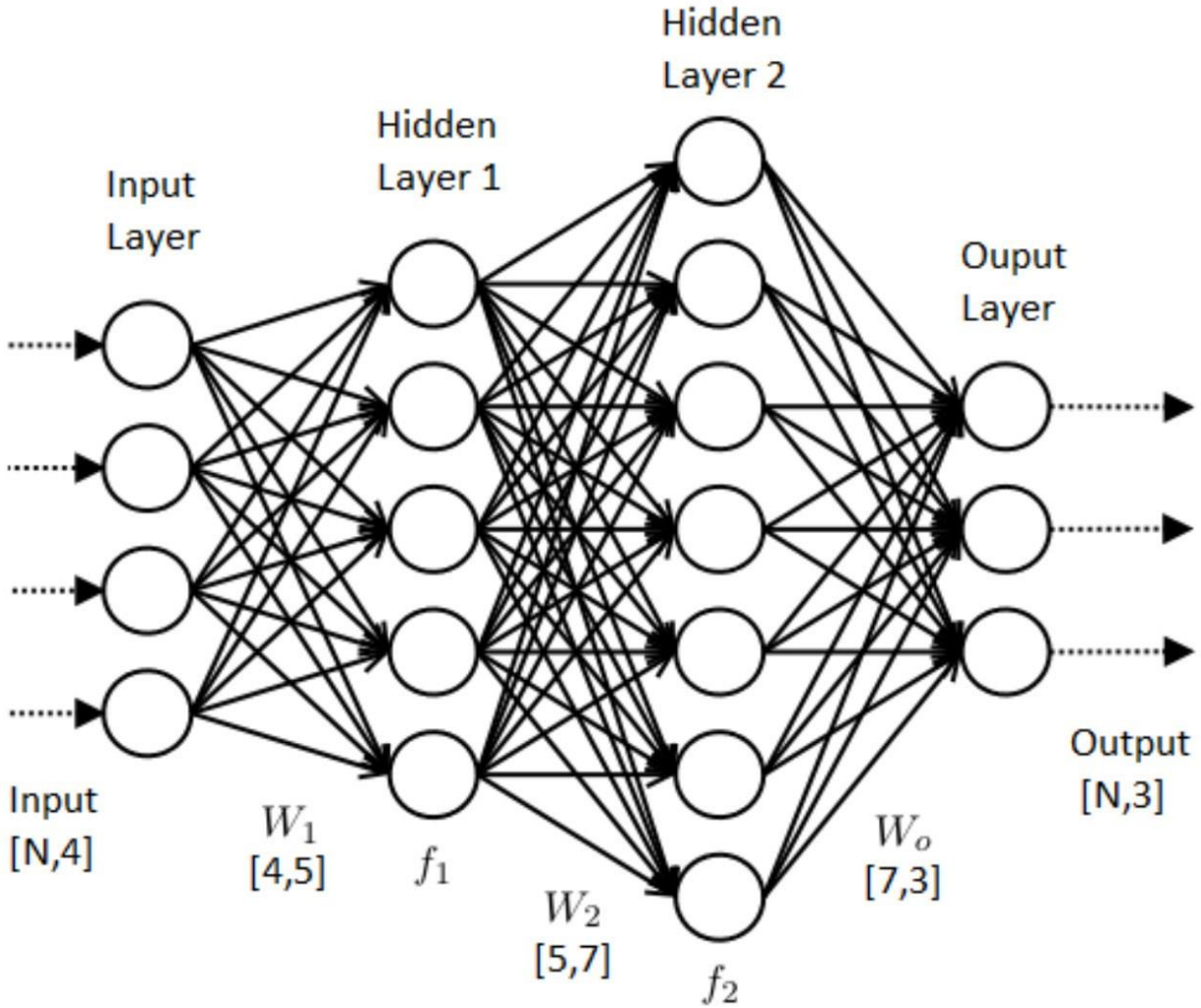
Aprendizado de Máquina
e Deep Learning

Rede neural com Keras

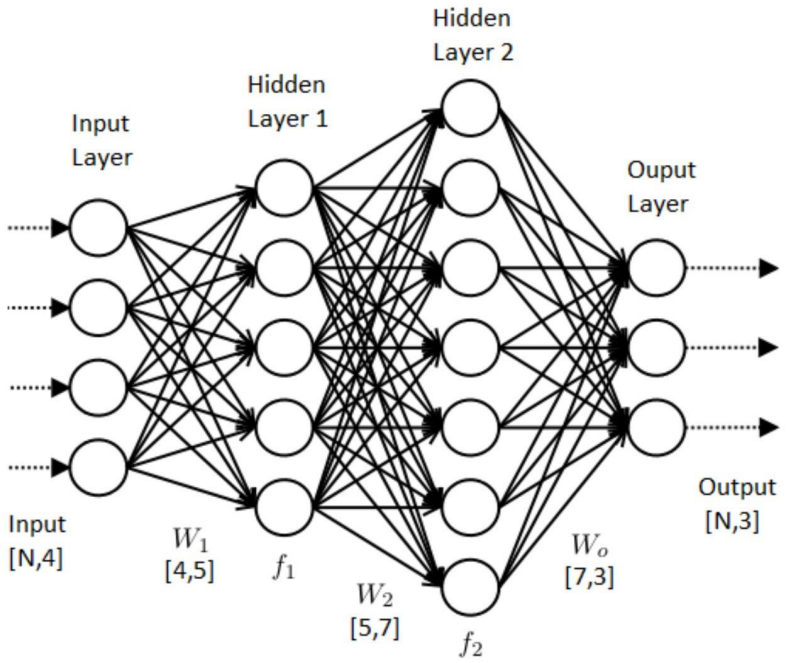
Prof. Dr. Thiago Meirelles Ventura

Rede neural artificial



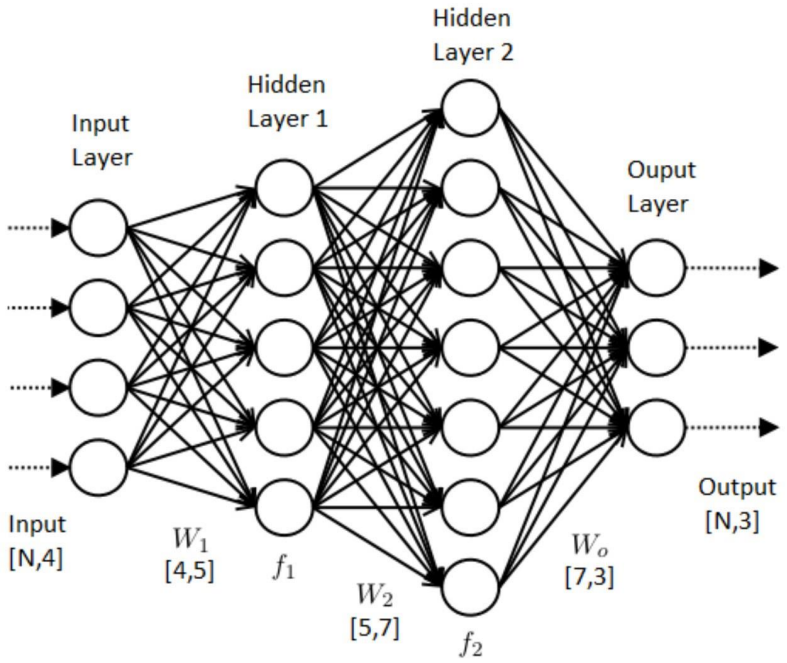
Rede neural artificial

- Camadas de uma RNA
  - Camda de entrada
  - Camadas intermediárias
  - Camada de saída



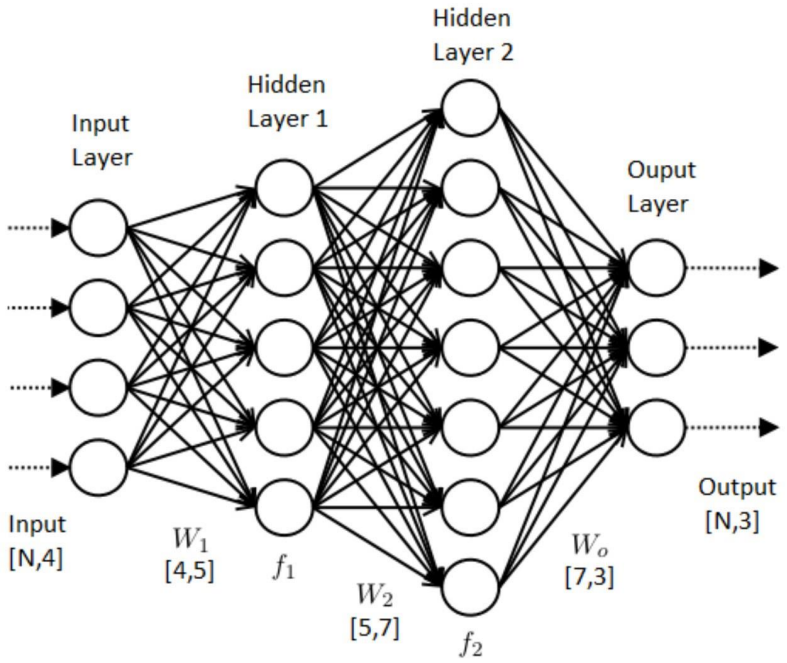
Rede neural artificial

- Principais componentes das camadas
  - Número de neurônios
  - Formato da entrada
  - Função de ativação



Rede neural artificial

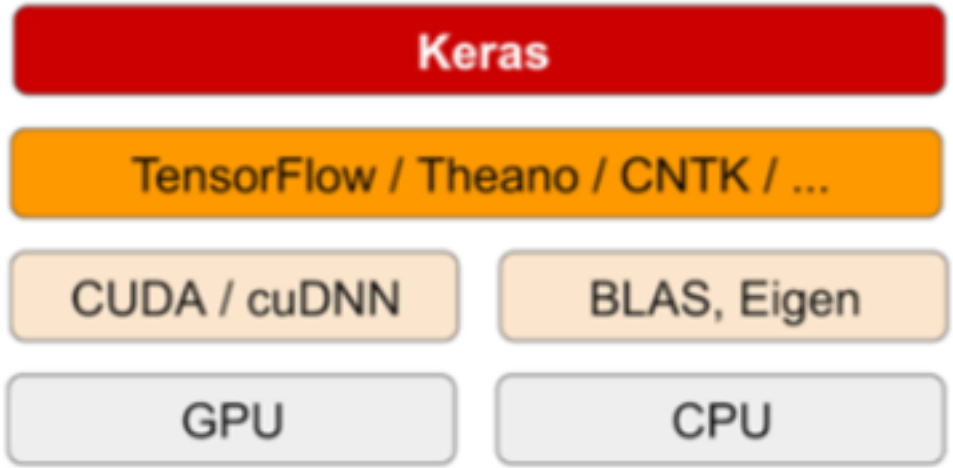
- Configuração da rede
  - Como a avaliação é feita
  - Como os pesos são atualizados




Keras

Keras

- *Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano*



Keras

- Facilita a construção, teste e uso de redes neurais
- Tenta deixar as etapas simples
  - Ao mesmo tempo que possibilita configurações detalhadas

Implementação

- Modelo

```
from keras.models import Sequential

model = Sequential()
```

Implementação

- Camadas

```
from keras.layers import Dense

model.add(Dense(units=10,
                activation='relu',
                input_dim=5))

model.add(Dense(units=3,
                activation='softmax'))
```

Exercício 5

- Desenhe a rede abaixo

```
model.add(Dense(units=10,
                activation='relu’,
                input_dim=5))
model.add(Dense(units=3,
                activation='softmax'))
```

Camadas no Keras

- Dense
- Activation
- Flatten
- Conv2D
- MaxPooling2D
- Dropout

Funções de ativação no Keras

- Linear
- Tanh
- Sigmoid
- Softmax
- Relu

Loss

- Avaliação de como estão as estimativas durante o treinamento
- Algumas funções de loss
  - mean_squared_error
  - mean_absolute_error
  - categorical_crossentropy
  - binary_crossentropy

Optmizers

- Forma de corrigir os pesos baseado no loss
- Alguns optmizers
  - SGD
  - RMSprop
  - Adam
  - Adagrad

Implementação

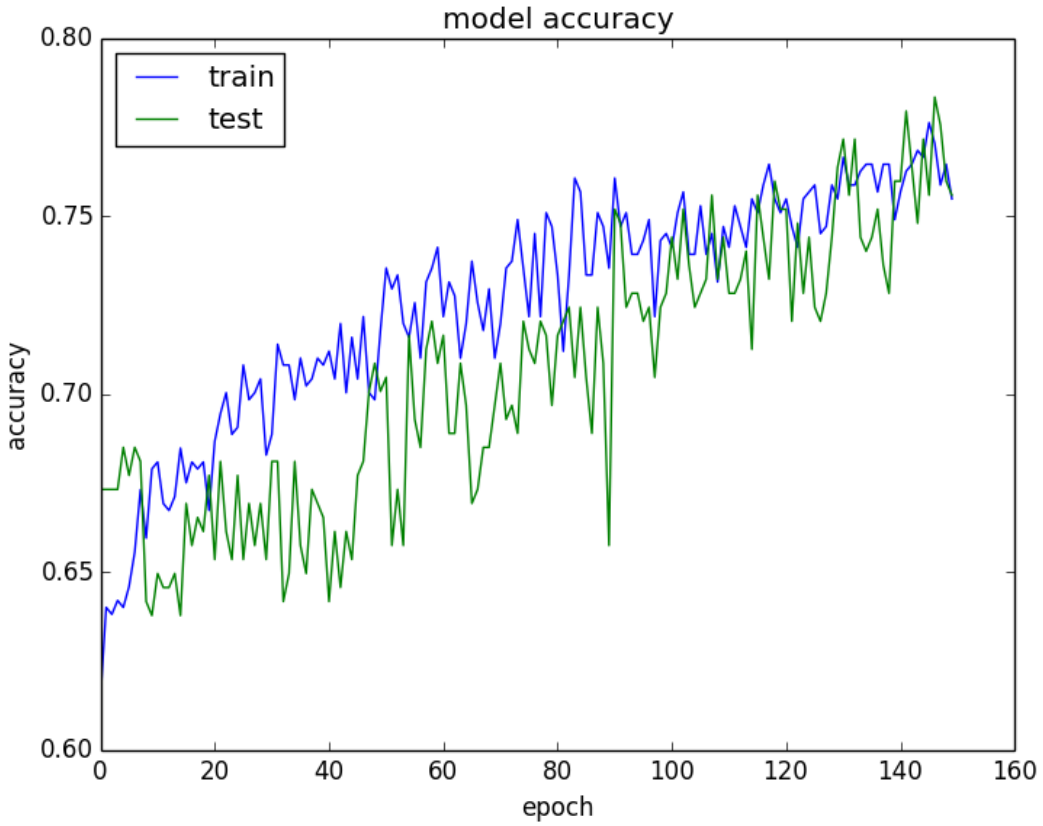
- Configuração do modelo

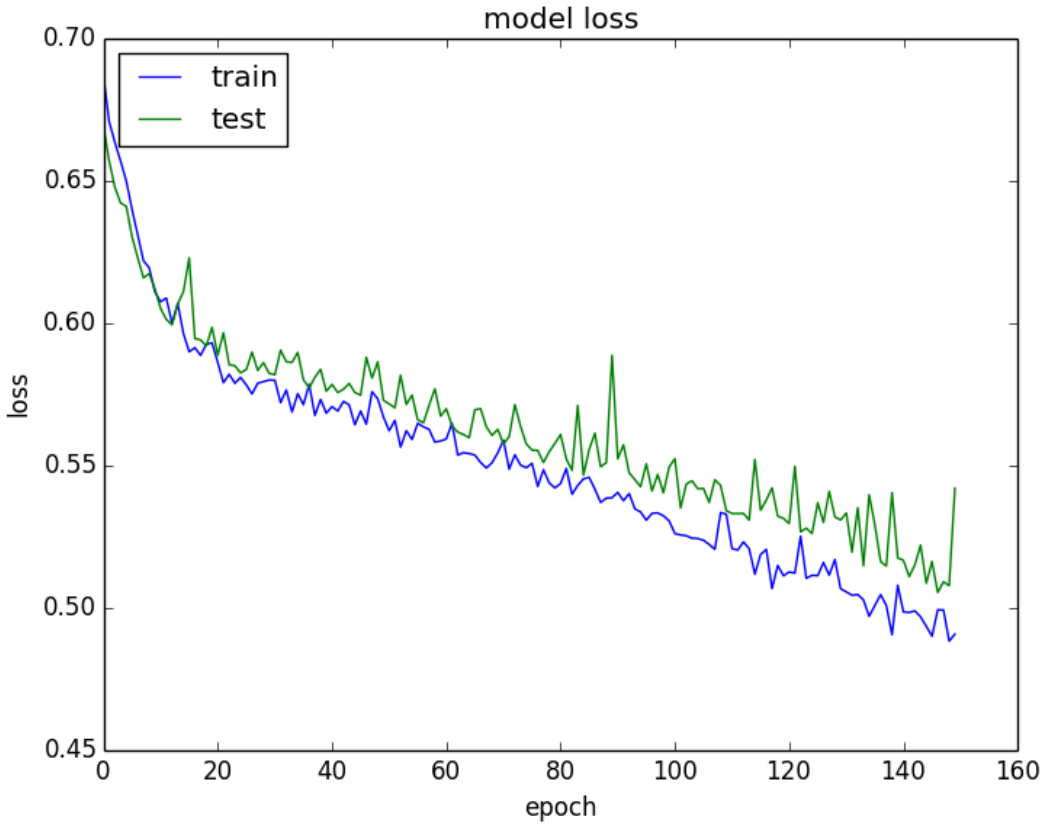
```
model.compile(loss='categorical_crossentropy',
              optimizer='sgd',
              metrics=['accuracy'])
```

Implementação

- Treinamento

```
history = model.fit(x_train,
                    y_train,
                    epochs=5,
                    batch_size=32)
```


model accuracy
train
test
accuracy
epoch
0.80
0.75
0.70
0.65
0.60
0
20
40
60
80
100
120
140
160


model loss
train
test
loss
epoch
0.70
0.65
0.60
0.55
0.50
0.45
0
20
40
60
80
100
120
140
160

Implementação

- Avaliação

```
loss_and_metrics = model.evaluate(x_test, y_test)
```

- Estimativa

```
classes = model.predict(x_test)
```

Exercício 6

- Criar uma rede neural que estima a nota de um aluno
  - Camada com 7 neurônios e função de ativação “tanh”
  - Camada com 3 neurônios e função de ativação “tanh”
  - Camada de saída com 1 neurônio e função de ativação “linear”
  - Optimizer SGD e e loss “mean_squared_error”
- A estimativa deve ser baseado em assiduidade, aproveitamento da monitoria, realização das listas de exercícios
  - Dados: notas.csv