“I need to sort out my wardrobe”

Deep Learning Demystified

Amy Boyd - Cloud Developer Advocate AI/ML
@AmyKateNicho

Microsoft

“Deep Learning is not as scary as I thought it was”

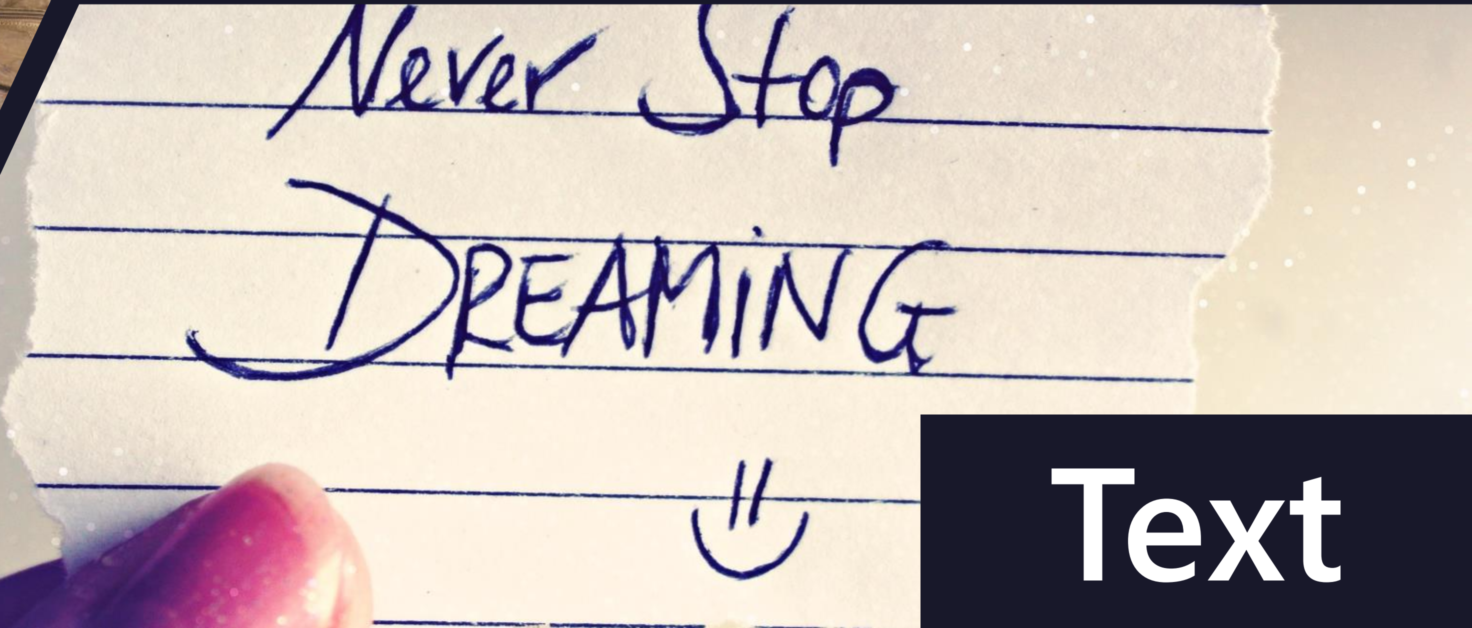
“I now feel I can hold a good conversation with a Data Scientist”

“I want to learn more and give it a go myself”

Sound

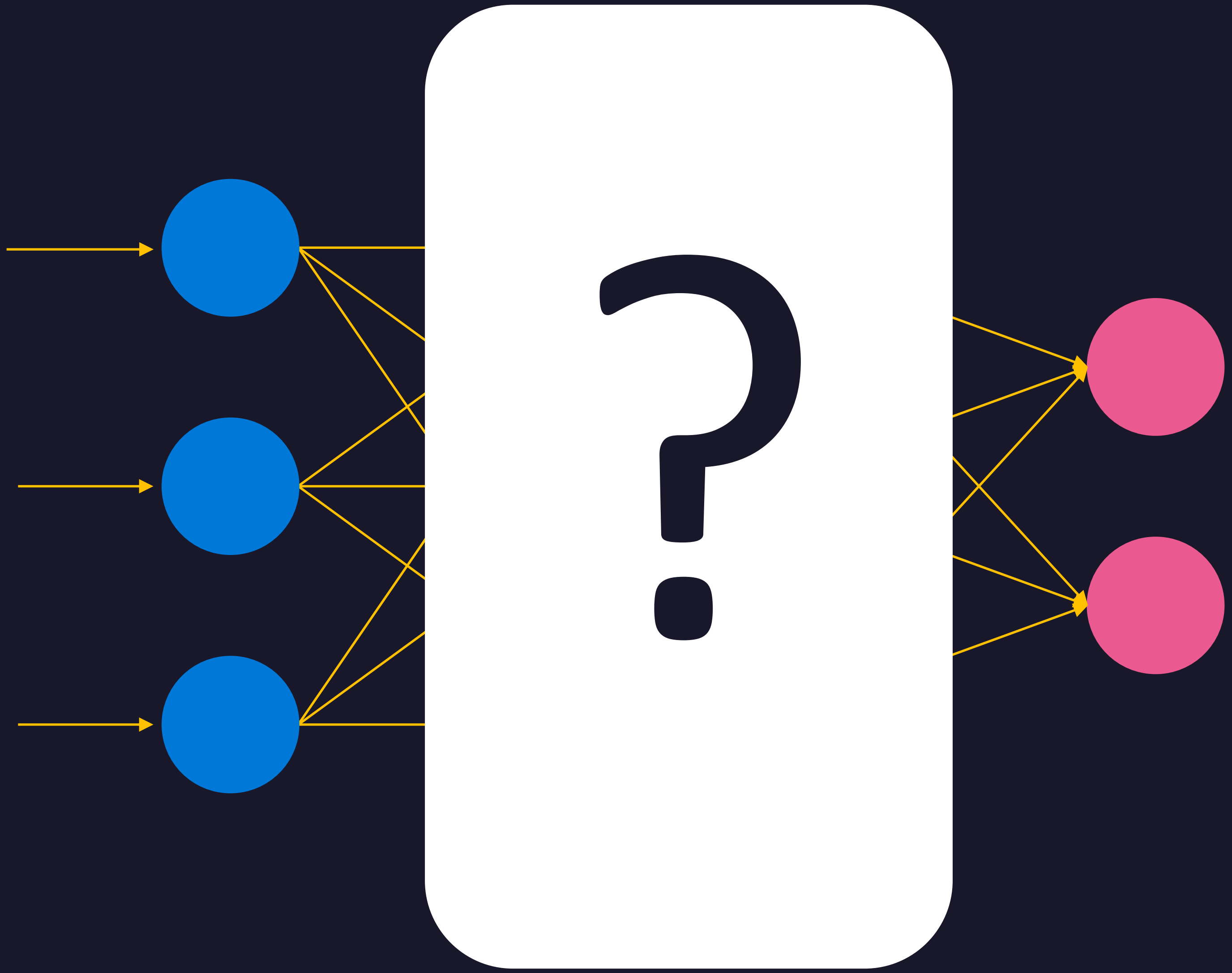
Video

Image

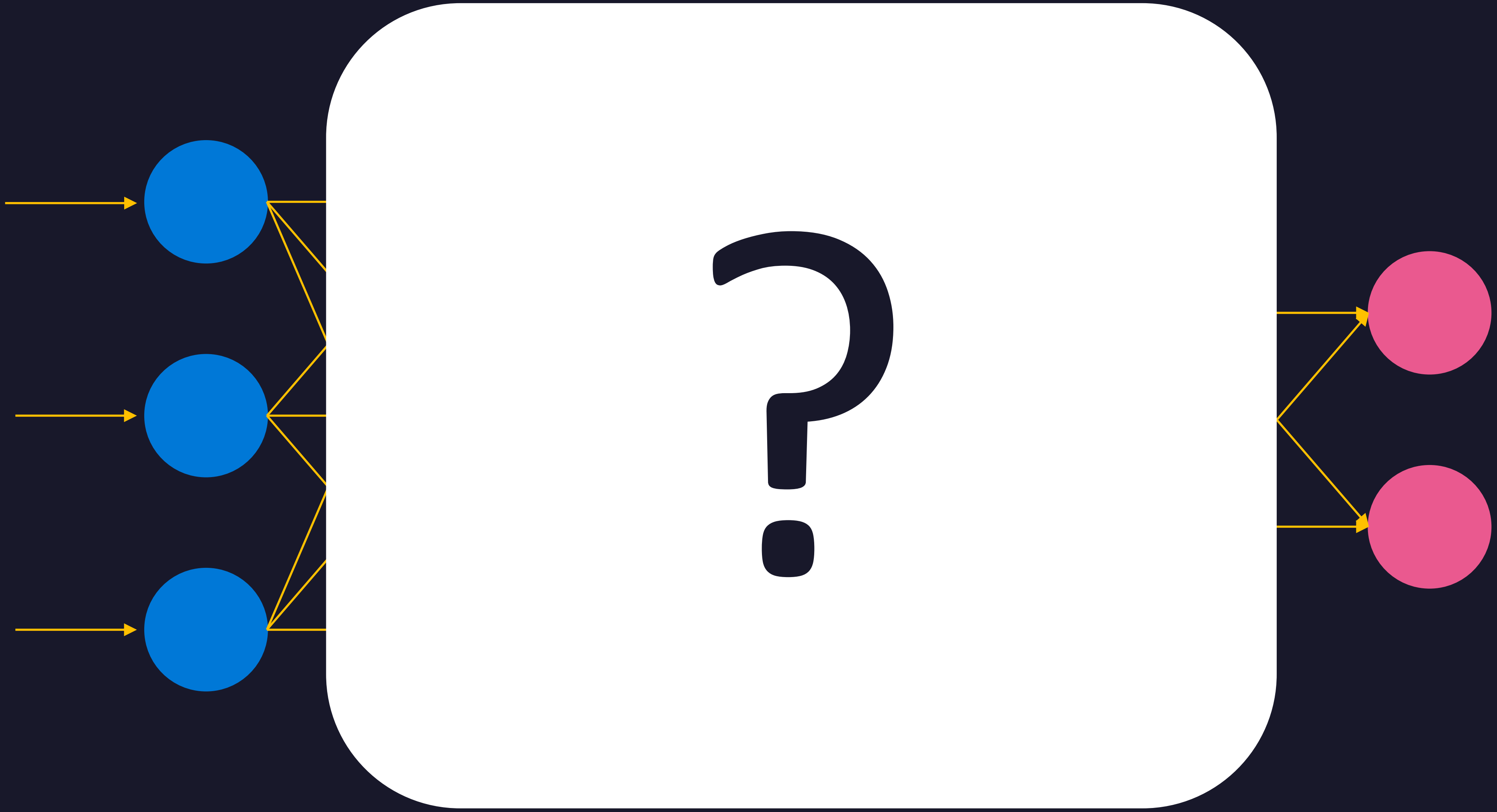


Text

Traditional ML: Neural Network



Deep Learning: Deep Neural Network



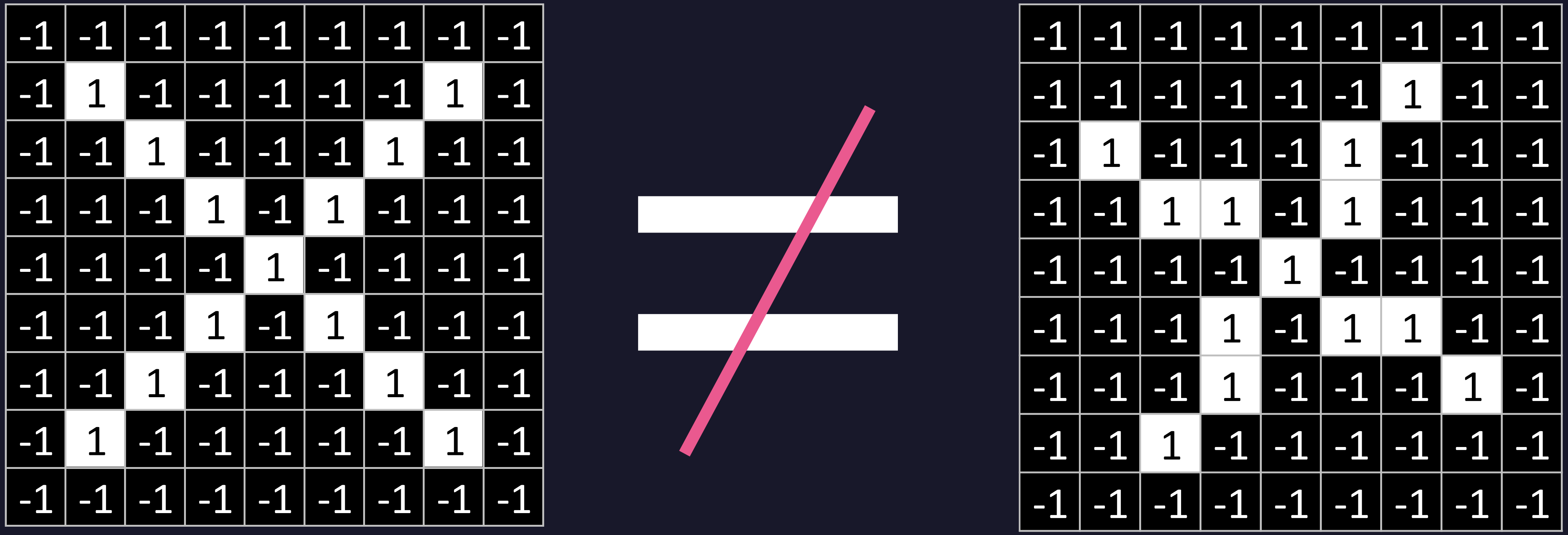
Sound

Image

Video

Text

Computers are so Literal!



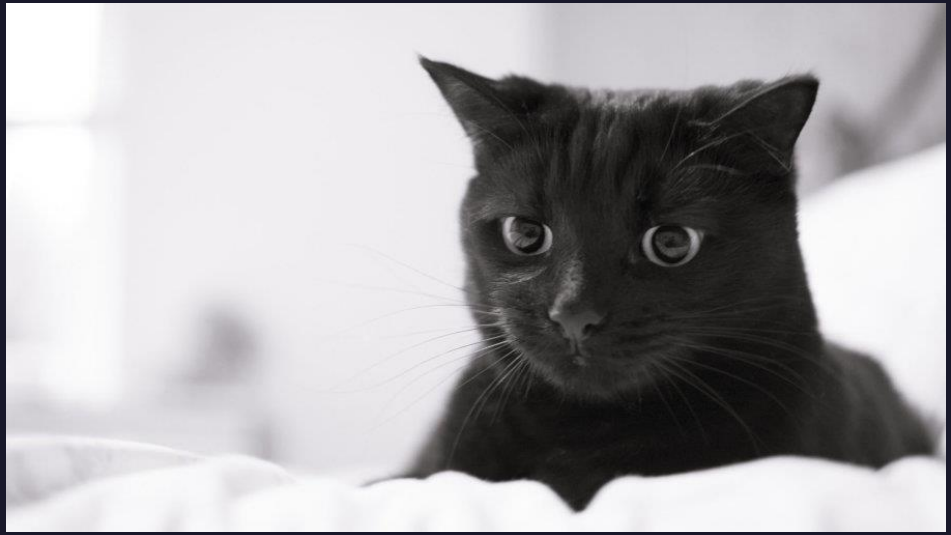
-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1

≠

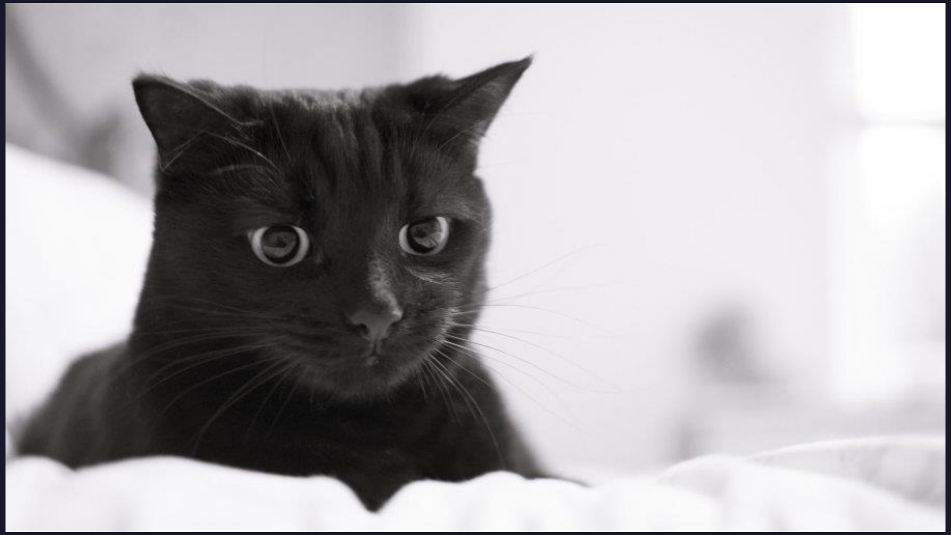
-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	1	-1	-1
-1	1	-1	-1	-1	1	-1	-1	-1
-1	-1	1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	1	-1	-1
-1	-1	-1	1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1

http://brohrer.github.io/how_convolutional_neural_networks_work.html

Computers are so Literal!



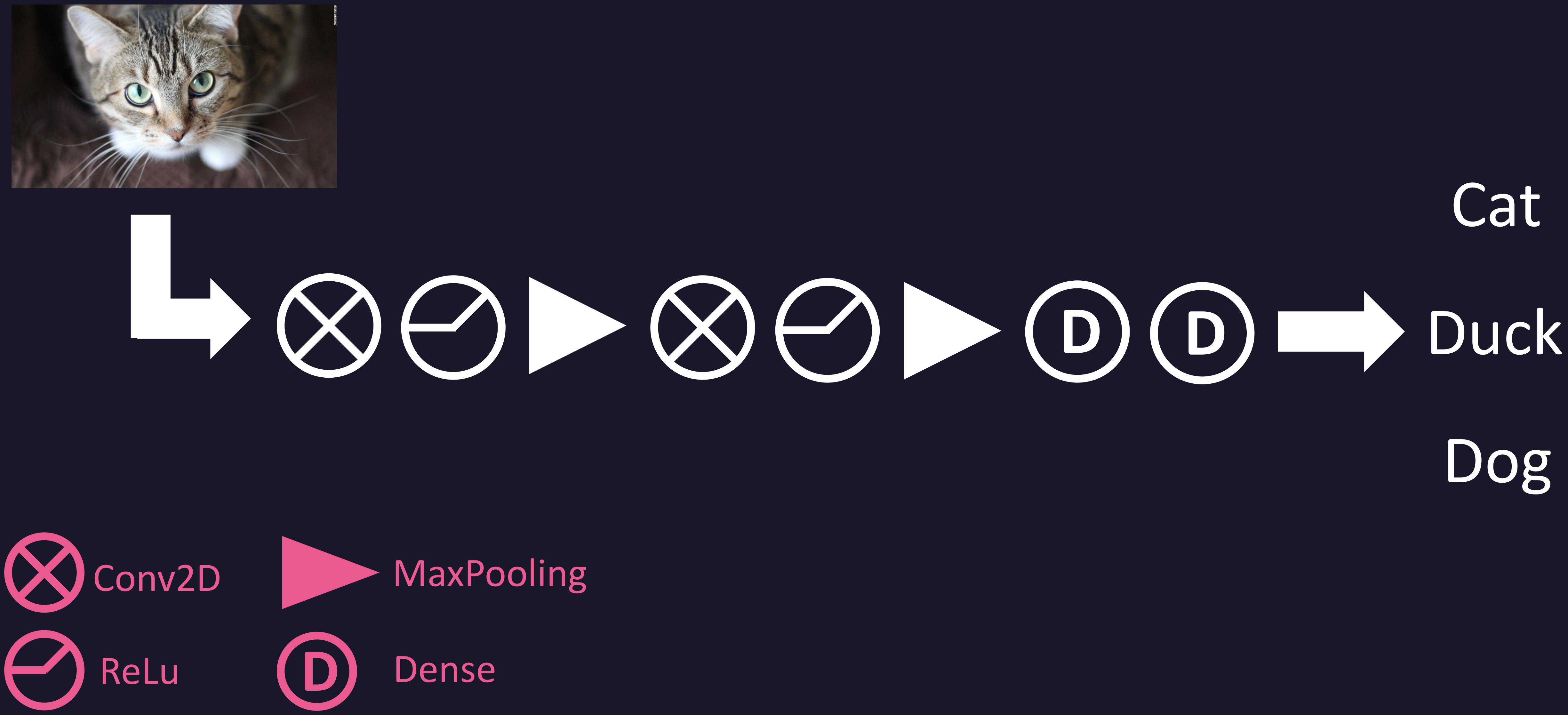
≠



Computers are so Literal!

≠

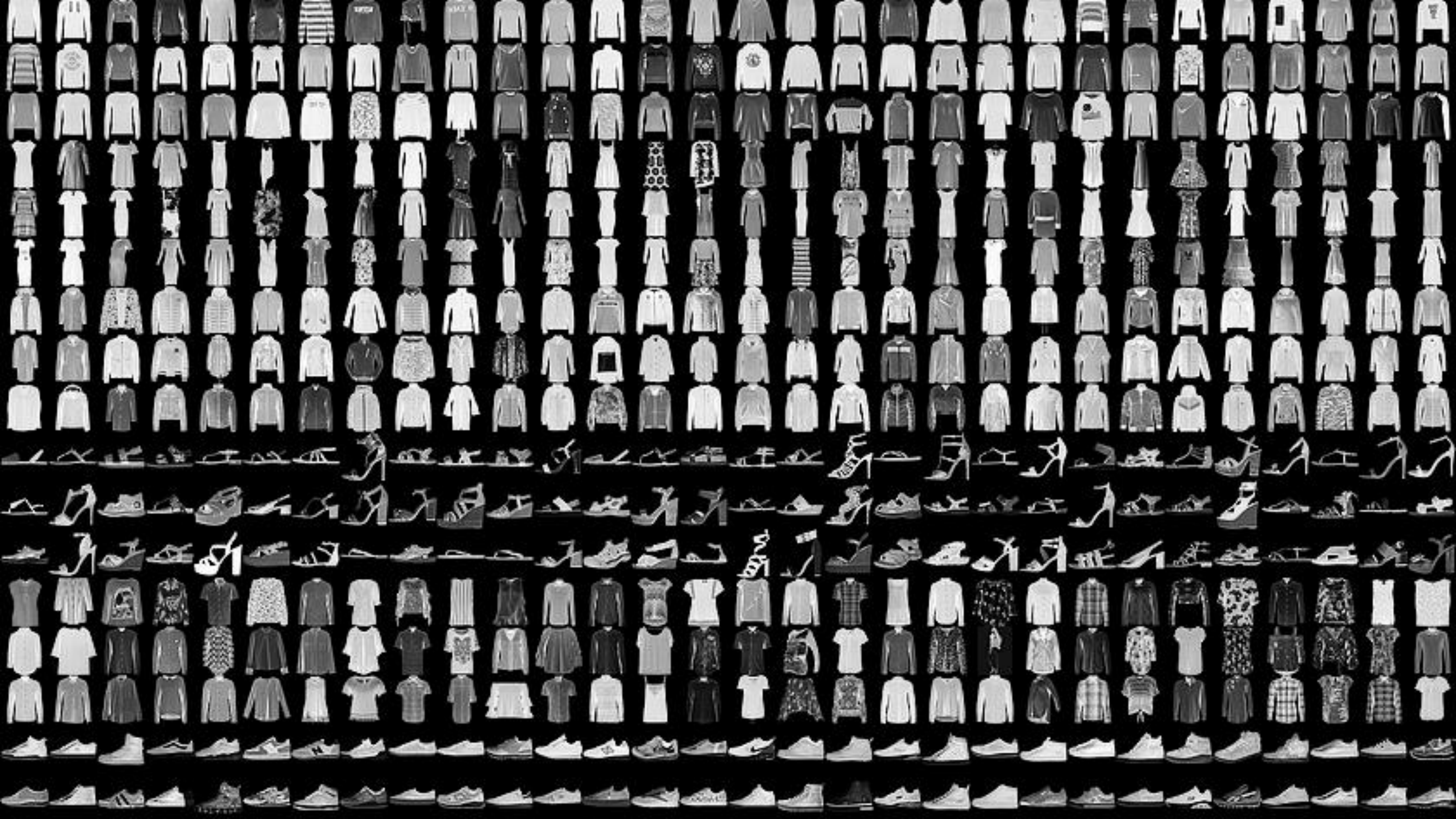
CNN: Convolutional Neural Network

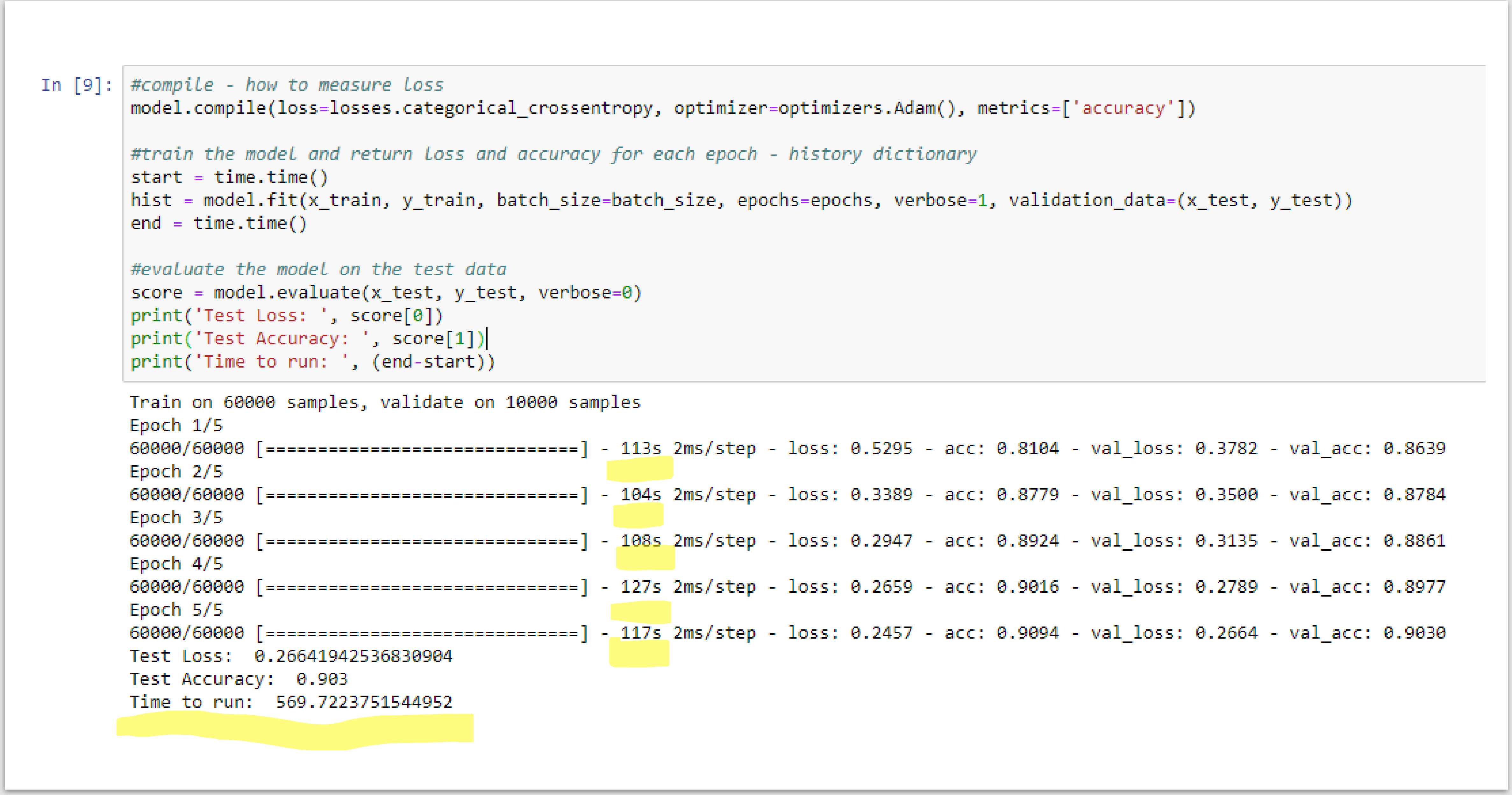


Zalando

Fashion-MNIST Dataset

https://github.com/zalandoresearch/fashion-mnist





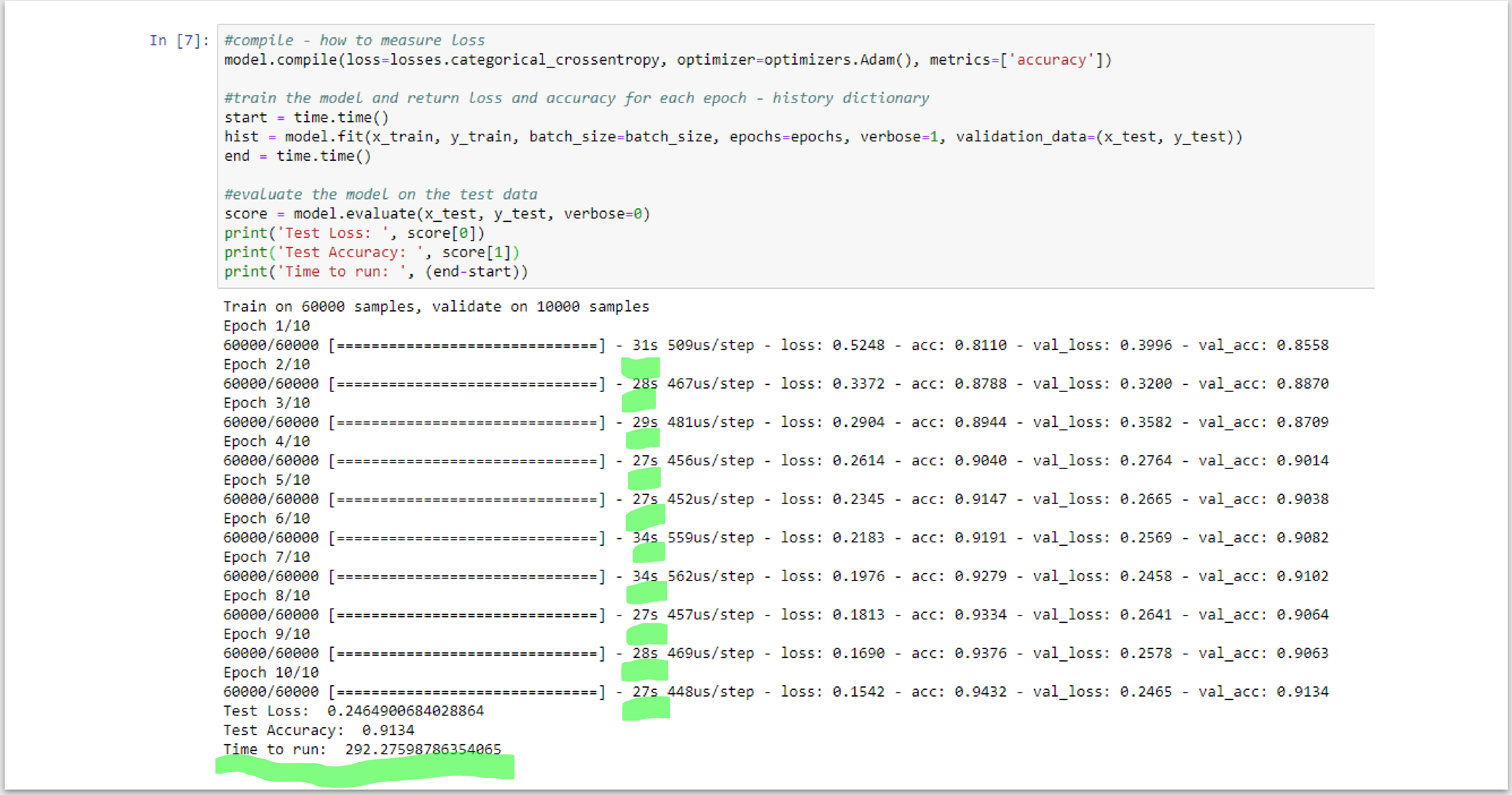
In [9]:

```
#compile - how to measure loss
model.compile(loss=losses.categorical_crossentropy, optimizer=optimizers.Adam(), metrics=['accuracy'])

#train the model and return loss and accuracy for each epoch - history dictionary
start = time.time()
hist = model.fit(x_train, y_train, batch_size=batch_size, epochs=epochs, verbose=1, validation_data=(x_test, y_test))
end = time.time()

#evaluate the model on the test data
score = model.evaluate(x_test, y_test, verbose=0)
print('Test Loss: ', score[0])
print('Test Accuracy: ', score[1])
print('Time to run: ', (end-start))
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/5
60000/60000 [==============================] - 113s 2ms/step - loss: 0.5295 - acc: 0.8104 - val_loss: 0.3782 - val_acc: 0.8639
Epoch 2/5
60000/60000 [==============================] - 104s 2ms/step - loss: 0.3389 - acc: 0.8779 - val_loss: 0.3500 - val_acc: 0.8784
Epoch 3/5
60000/60000 [==============================] - 108s 2ms/step - loss: 0.2947 - acc: 0.8924 - val_loss: 0.3135 - val_acc: 0.8861
Epoch 4/5
60000/60000 [==============================] - 127s 2ms/step - loss: 0.2659 - acc: 0.9016 - val_loss: 0.2789 - val_acc: 0.8977
Epoch 5/5
60000/60000 [==============================] - 117s 2ms/step - loss: 0.2457 - acc: 0.9094 - val_loss: 0.2664 - val_acc: 0.9030
Test Loss:  0.26641942536830904
Test Accuracy:  0.903
Time to run:  569.7223751544952
```



In [7]:

```
#compile - how to measure loss
model.compile(loss=losses.categorical_crossentropy, optimizer=optimizers.Adam(), metrics=['accuracy'])

#train the model and return loss and accuracy for each epoch - history dictionary
start = time.time()
hist = model.fit(x_train, y_train, batch_size=batch_size, epochs=epochs, verbose=1, validation_data=(x_test, y_test))
end = time.time()

#evaluate the model on the test data
score = model.evaluate(x_test, y_test, verbose=0)
print('Test Loss: ', score[0])
print('Test Accuracy: ', score[1])
print('Time to run: ', (end-start))
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/10
60000/60000 [==============================] - 31s 509us/step - loss: 0.5248 - acc: 0.8110 - val_loss: 0.3996 - val_acc: 0.8558
Epoch 2/10
60000/60000 [==============================] - 28s 467us/step - loss: 0.3372 - acc: 0.8788 - val_loss: 0.3200 - val_acc: 0.8870
Epoch 3/10
60000/60000 [==============================] - 29s 481us/step - loss: 0.2904 - acc: 0.8944 - val_loss: 0.3582 - val_acc: 0.8709
Epoch 4/10
60000/60000 [==============================] - 27s 456us/step - loss: 0.2614 - acc: 0.9040 - val_loss: 0.2764 - val_acc: 0.9014
Epoch 5/10
60000/60000 [==============================] - 27s 452us/step - loss: 0.2345 - acc: 0.9147 - val_loss: 0.2665 - val_acc: 0.9038
Epoch 6/10
60000/60000 [==============================] - 34s 559us/step - loss: 0.2183 - acc: 0.9191 - val_loss: 0.2569 - val_acc: 0.9082
Epoch 7/10
60000/60000 [==============================] - 34s 562us/step - loss: 0.1976 - acc: 0.9279 - val_loss: 0.2458 - val_acc: 0.9102
Epoch 8/10
60000/60000 [==============================] - 27s 457us/step - loss: 0.1813 - acc: 0.9334 - val_loss: 0.2641 - val_acc: 0.9064
Epoch 9/10
60000/60000 [==============================] - 28s 469us/step - loss: 0.1690 - acc: 0.9376 - val_loss: 0.2578 - val_acc: 0.9063
Epoch 10/10
60000/60000 [==============================] - 27s 448us/step - loss: 0.1542 - acc: 0.9432 - val_loss: 0.2465 - val_acc: 0.9134
Test Loss:  0.24649006840288864
Test Accuracy:  0.9134
Time to run:  292.27598786354065
```

*** free compute*

CPU

Avg = ~114 secs

Time to run per epoch. 5 epochs run

*** NC6 series*

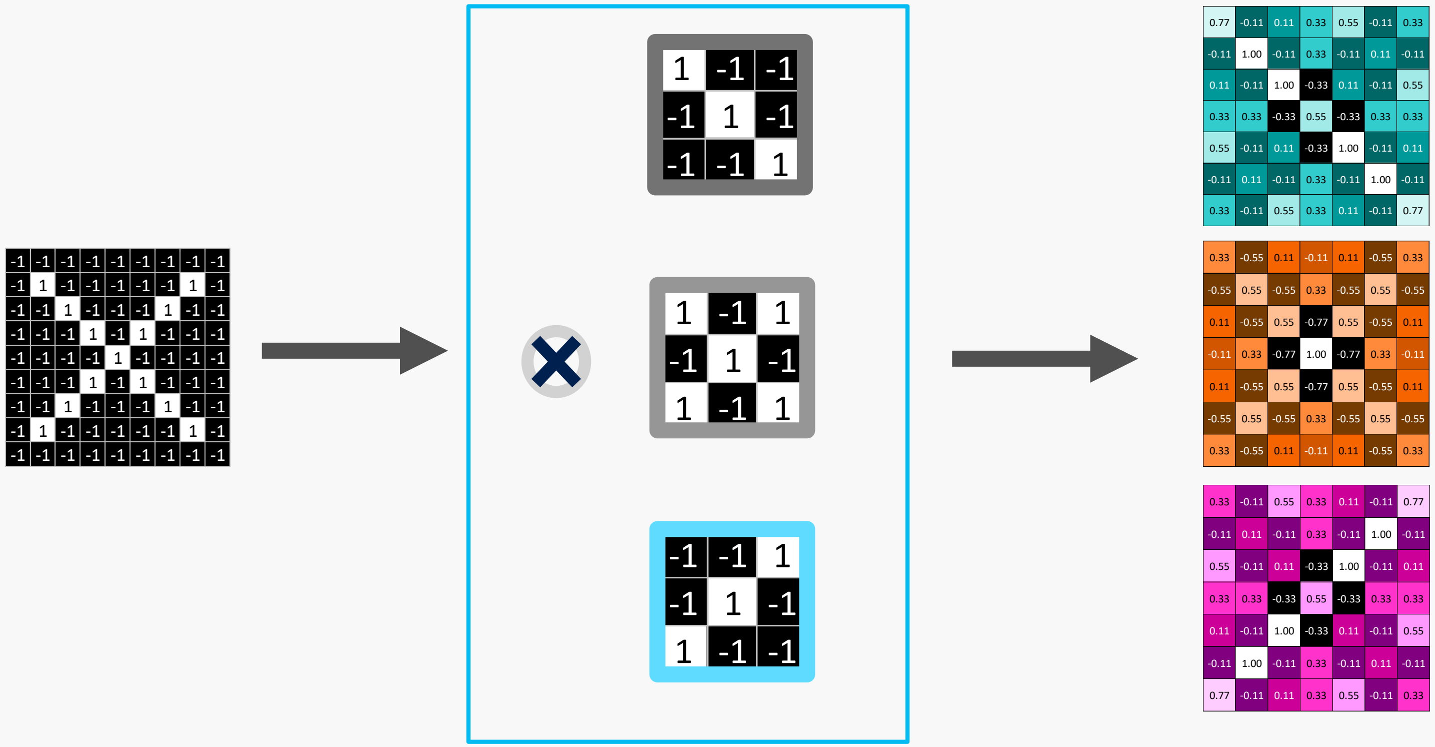
GPU

Avg = ~29 secs

Time to run per epoch. 10 epochs run

Convolution layer

One image becomes a stack of filtered images



-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1

1	-1	-1
-1	1	-1
-1	-1	1

1	-1	1
-1	1	-1
1	-1	1

-1	-1	1
-1	1	-1
1	-1	-1

0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

0.33	-0.55	0.11	-0.11	0.11	-0.55	0.33
-0.55	0.55	-0.55	0.33	-0.55	0.55	-0.55
0.11	-0.55	0.55	-0.77	0.55	-0.55	0.11
-0.11	0.33	-0.77	1.00	-0.77	0.33	-0.11
0.11	-0.55	0.55	-0.77	0.55	-0.55	0.11
-0.55	0.55	-0.55	0.33	-0.55	0.55	-0.55
0.33	-0.55	0.11	-0.11	0.11	-0.55	0.33

0.33	-0.11	0.55	0.33	0.11	-0.11	0.77
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.77	-0.11	0.11	0.33	0.55	-0.11	0.33

http://brohrer.github.io/how_convolutional_neural_networks_work.html

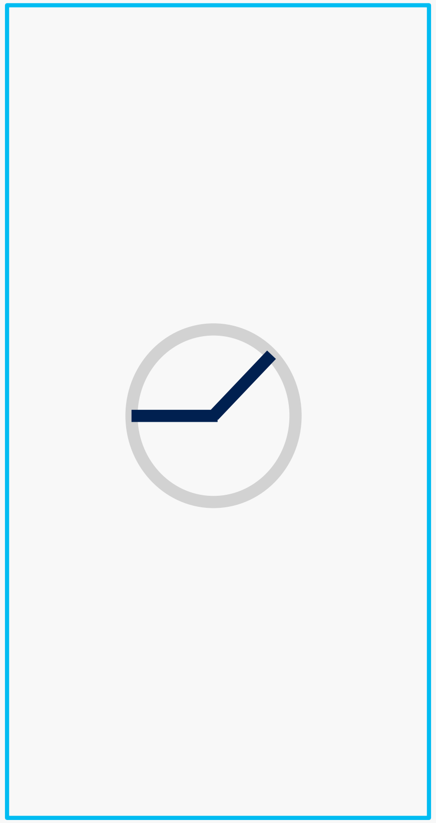
ReLU layer

A stack of images becomes a stack of images with no negative values.

0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

0.33	-0.55	0.11	-0.11	0.11	-0.55	0.33
-0.55	0.55	-0.55	0.33	-0.55	0.55	-0.55
0.11	-0.55	0.55	-0.77	0.55	-0.55	0.11
-0.11	0.33	-0.77	1.00	-0.77	0.33	-0.11
0.11	-0.55	0.55	-0.77	0.55	-0.55	0.11
-0.55	0.55	-0.55	0.33	-0.55	0.55	-0.55
0.33	-0.55	0.11	-0.11	0.11	-0.55	0.33

0.33	-0.11	0.55	0.33	0.11	-0.11	0.77
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.77	-0.11	0.11	0.33	0.55	-0.11	0.33



0.77	0	0.11	0.33	0.55	0	0.33
0	1.00	0	0.33	0	0.11	0
0.11	0	1.00	0	0.11	0	0.55
0.33	0.33	0	0.55	0	0.33	0.33
0.55	0	0.11	0	1.00	0	0.11
0	0.11	0	0.33	0	1.00	0
0.33	0	0.55	0.33	0.11	0	0.77

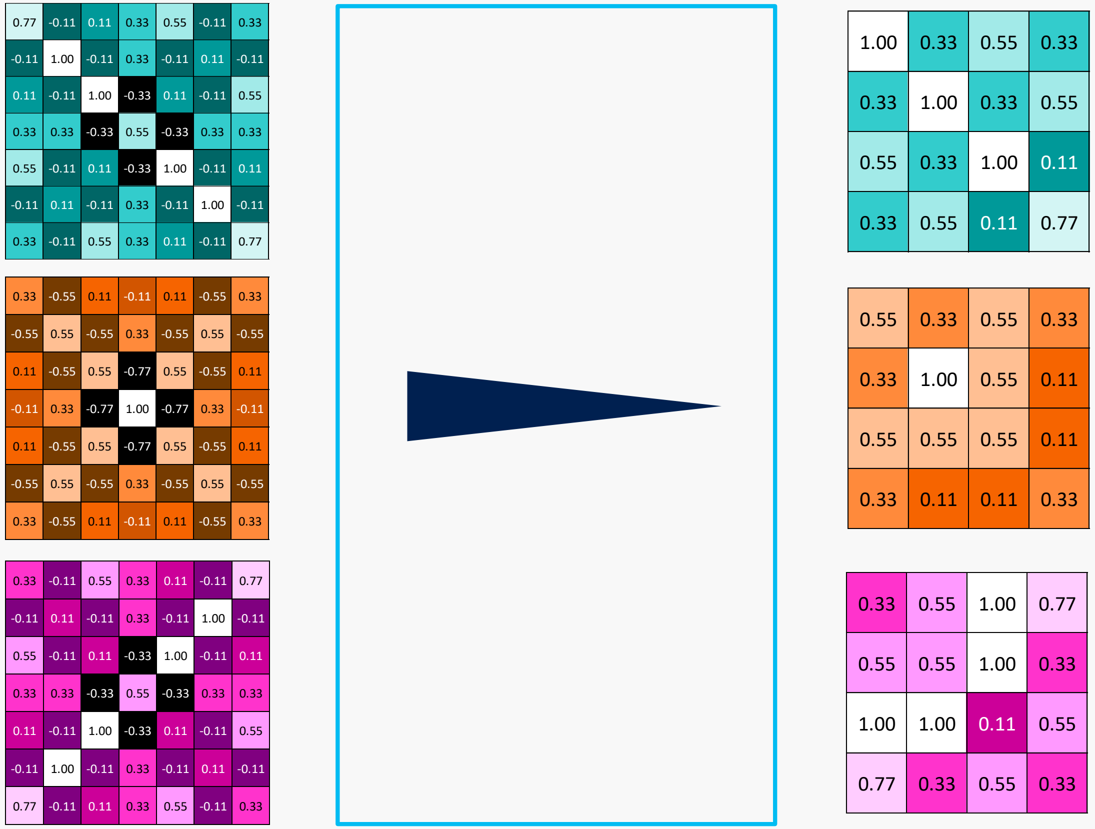
0.33	0	0.11	0	0.11	0	0.33
0	0.55	0	0.33	0	0.55	0
0.11	0	0.55	0	0.55	0	0.11
0	0.33	0	1.00	0	0.33	0
0.11	0	0.55	0	0.55	0	0.11
0	0.55	0	0.33	0	0.55	0
0.33	0	0.11	0	0.11	0	0.33

0.33	0	0.55	0.33	0.11	0	0.77
0	0.11	0	0.33	0	1.00	0
0.55	0	0.11	0	1.00	0	0.11
0.33	0.33	0	0.55	0	0.33	0.33
0.11	0	1.00	0	0.11	0	0.55
0	1.00	0	0.33	0	0.11	0
0.77	0	0.11	0.33	0.55	0	0.33

http://brohrer.github.io/how_convolutional_neural_networks_work.html

Pooling layer

A stack of images becomes a stack of smaller images.



0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

0.33	-0.55	0.11	-0.11	0.11	-0.55	0.33
-0.55	0.55	-0.55	0.33	-0.55	0.55	-0.55
0.11	-0.55	0.55	-0.77	0.55	-0.55	0.11
-0.11	0.33	-0.77	1.00	-0.77	0.33	-0.11
0.11	-0.55	0.55	-0.77	0.55	-0.55	0.11
-0.55	0.55	-0.55	0.33	-0.55	0.55	-0.55
0.33	-0.55	0.11	-0.11	0.11	-0.55	0.33

0.33	-0.11	0.55	0.33	0.11	-0.11	0.77
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.77	-0.11	0.11	0.33	0.55	-0.11	0.33

1.00	0.33	0.55	0.33
0.33	1.00	0.33	0.55
0.55	0.33	1.00	0.11
0.33	0.55	0.11	0.77

0.55	0.33	0.55	0.33
0.33	1.00	0.55	0.11
0.55	0.55	0.55	0.11
0.33	0.11	0.11	0.33

0.33	0.55	1.00	0.77
0.55	0.55	1.00	0.33
1.00	1.00	0.11	0.55
0.77	0.33	0.55	0.33

http://brohrer.github.io/how_convolutional_neural_networks_work.html

Deep Learning: Key Takeaways

- Deep Learning is not as scary as I thought it was
- I now feel I can hold a good conversation with a Data Scientist
- I want to learn more and give it a go myself

Deep Learning: Key Takeaways

- Azure Notebooks: https://docs.microsoft.com/en-us/azure/notebooks/
- Deep learning with Keras Course: https://app.pluralsight.com/library/courses/keras-deep-learning/table-of-contents
- Keras API Documentation: https://keras.io/

“I need to sort out my wardrobe”

Deep Learning Demystified

Amy Boyd - Cloud Developer Advocate AI/ML
@AmyKateNicho

Microsoft