Learning from zero...

Some issues you might face

Laurence Moroney, Google

Steps to take

1. Get as many examples of shoes as possible
2. Train using these examples
3. Profit!

na
višjem

Steps to take

1. Get as many examples of shoes as possible
2. Train using these examples
3. Profit!

```
Training accuracy: .920
Training accuracy: .935
Training accuracy: .947
Training accuracy: .961
Training accuracy: .977
Training accuracy: .995
Training accuracy: 1.00
```

Steps to take

1. ~~Get as many examples of shoes as possible~~
2. ~~Train using these examples~~
3. Profit?

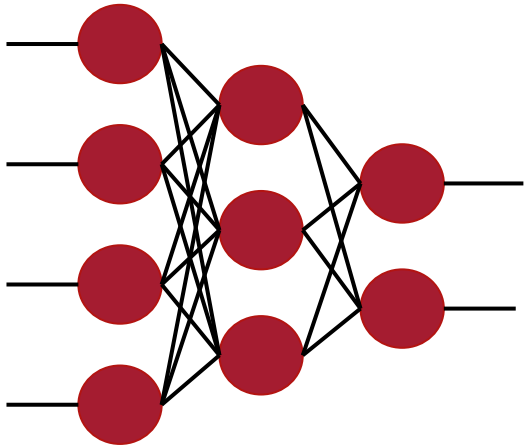
Data

The network ‘sees’ everything. Has no context for measuring how well it does with data it has never previously been exposed to.

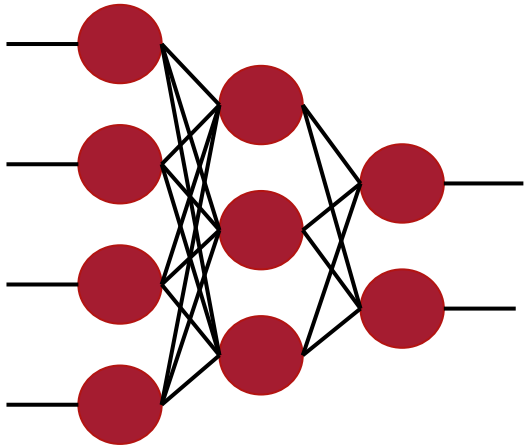
The network ‘sees’ a subset of your data. You can use the rest to measure its performance against previously unseen data.

The network ‘sees’ a subset of your data. You can use an unseen subset to measure its accuracy while training (validation), and then another subset to measure its accuracy after it’s finished training (test).

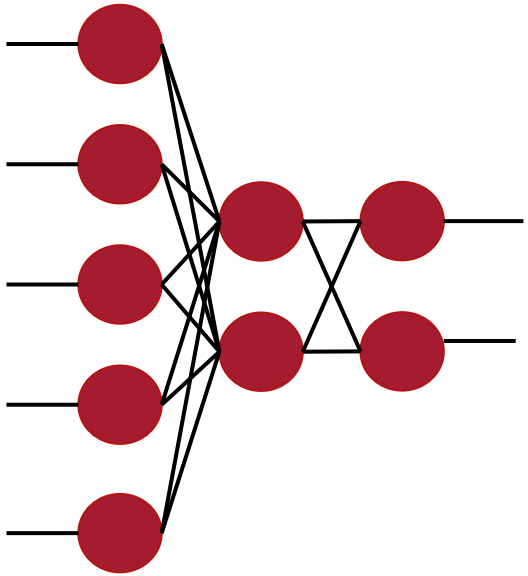
Data	Validation Data	Test Data
Accuracy: 0.999	Accuracy: 0.920	Accuracy: 0.800



Data	Validation Data	Test Data
Accuracy: 0.999	Accuracy: 0.920	Accuracy: 0.800



Data	Validation Data	Test Data
Accuracy: 0.942	Accuracy: 0.930	Accuracy: 0.925



```
import tensorflow as tf

data = tf.keras.datasets.mnist
(training_images, training_labels), (val_images, val_labels) = data.load_data()

training_images  = training_images / 255.0
val_images = val_images / 255.0

model = tf.keras.models.Sequential(
    [tf.keras.layers.Flatten(input_shape=(28,28)),
     tf.keras.layers.Dense(20, activation=tf.nn.relu),
     tf.keras.layers.Dense(10, activation=tf.nn.softmax)])
```

```
model.fit(training_images, training_labels, epochs=20)
```

```
Epoch 16/20
1875/1875 [==============================] - 3s 2ms/step - loss: 0.0931 - accuracy: 0.9729
Epoch 17/20
1875/1875 [==============================] - 3s 2ms/step - loss: 0.0907 - accuracy: 0.9731
Epoch 18/20
1875/1875 [==============================] - 3s 2ms/step - loss: 0.0892 - accuracy: 0.9735
Epoch 19/20
1875/1875 [==============================] - 3s 2ms/step - loss: 0.0868 - accuracy: 0.9740
Epoch 20/20
1875/1875 [==============================] - 3s 2ms/step - loss: 0.0844 - accuracy: 0.9750
```

```
model.fit(training_images, training_labels,
          validation_data=(val_images, val_labels),
          epochs=20)
```

```
model.fit(training_images, training_labels,
          validation_data=(val_images, val_labels),
          epochs=20)
```

```
Epoch 14/20
1875/1875 [==============================] - 4s 2ms/step - loss: 0.0960 - accuracy: 0.9713 - val_loss: 0.1410 - val_accuracy: 0.9588
Epoch 15/20
1875/1875 [==============================] - 4s 2ms/step - loss: 0.0933 - accuracy: 0.9722 - val_loss: 0.1353 - val_accuracy: 0.9626
Epoch 16/20
1875/1875 [==============================] - 4s 2ms/step - loss: 0.0910 - accuracy: 0.9729 - val_loss: 0.1356 - val_accuracy: 0.9622
Epoch 17/20
1875/1875 [==============================] - 4s 2ms/step - loss: 0.0880 - accuracy: 0.9734 - val_loss: 0.1339 - val_accuracy: 0.9625
Epoch 18/20
1875/1875 [==============================] - 4s 2ms/step - loss: 0.0855 - accuracy: 0.9740 - val_loss: 0.1349 - val_accuracy: 0.9619
Epoch 19/20
1875/1875 [==============================] - 4s 2ms/step - loss: 0.0832 - accuracy: 0.9754 - val_loss: 0.1341 - val_accuracy: 0.9624
Epoch 20/20
1875/1875 [==============================] - 4s 2ms/step - loss: 0.0808 - accuracy: 0.9759 - val_loss: 0.1340 - val_accuracy: 0.9626
```

```
model.evaluate(test_images, test_labels)
```

```
313/313 [==============================] - 1s 2ms/step - loss: 0.1495 - accuracy: 0.9569
```

Quiz!