

Copyright Notice

These slides are distributed under the Creative Commons License.

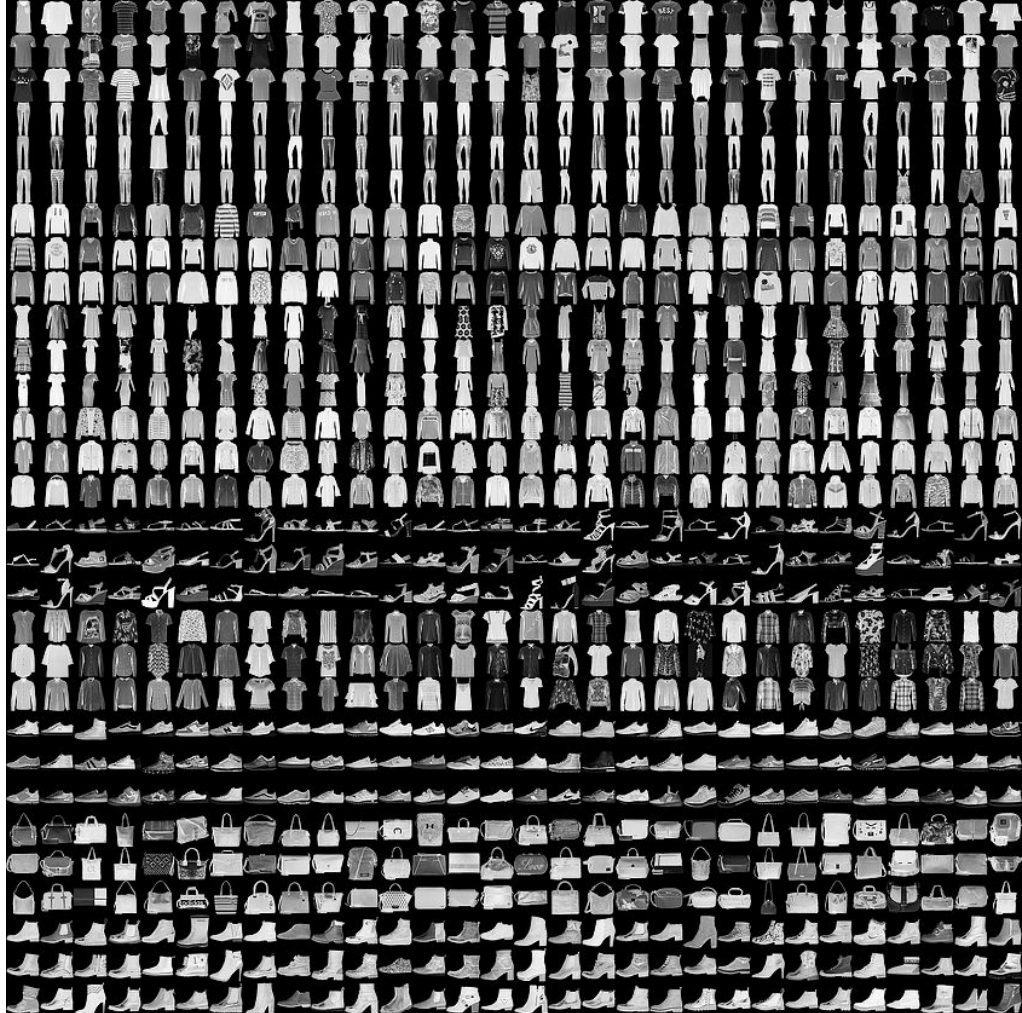
[DeepLearning.AI](#) makes these slides available for educational purposes. You may not use or distribute these slides for commercial purposes. You may make copies of these slides and use or distribute them for educational purposes as long as you cite [DeepLearning.AI](#) as the source of the slides.

For the rest of the details of the license, see <https://creativecommons.org/licenses/by-sa/2.0/legalcode>



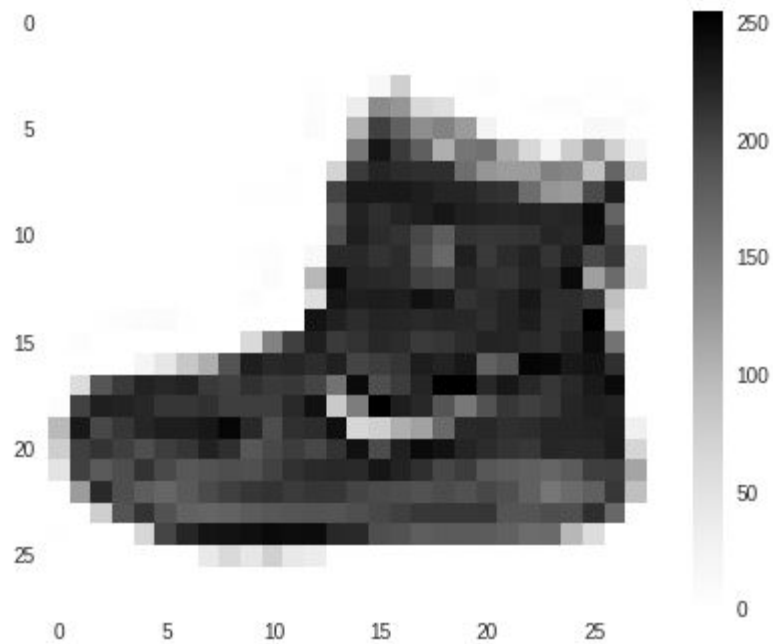
Fashion MNIST

- 70k Images
- 10 Categories
- Images are 28x28
- Can train a neural net!



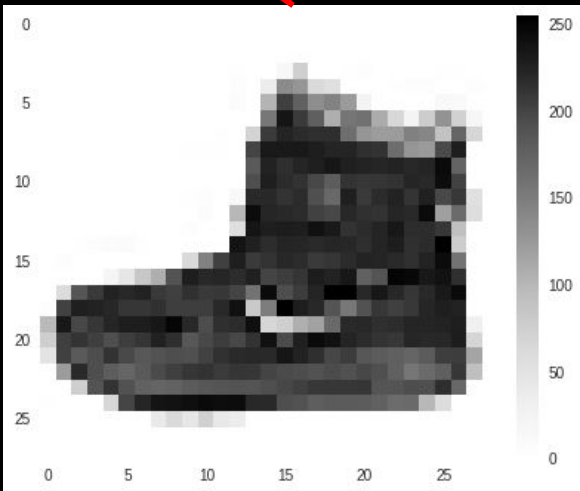
Fashion MNIST

- 70k Images
- 10 Categories
- Images are 28x28
- Can train a neural net!



```
fashion_mnist = tf.keras.datasets.fashion_mnist  
(train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data()
```

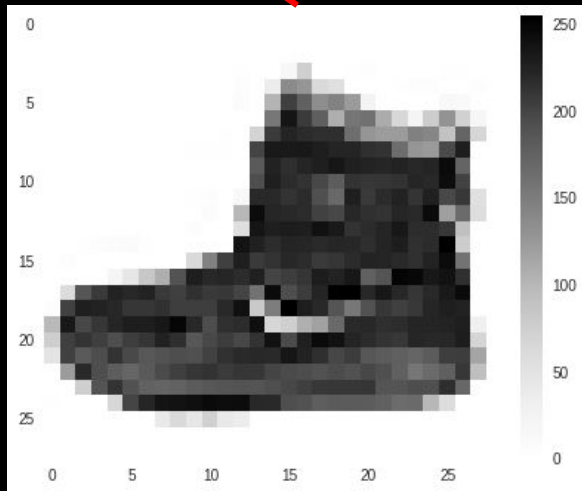
```
fashion_mnist = tf.keras.datasets.fashion_mnist  
(train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data()
```



09

```
import tensorflow as tf
from tensorflow import keras
```

```
mnist = tf.keras.datasets.fashion_mnist
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()
```



09

09 = ankle boot;
踝靴;
アンクルブーツ;
Bróg rúitín

```
model = keras.Sequential([  
    keras.layers.Flatten(),  
    keras.layers.Dense(128, activation=tf.nn.relu),  
    keras.layers.Dense(10, activation=tf.nn.softmax)  
])
```

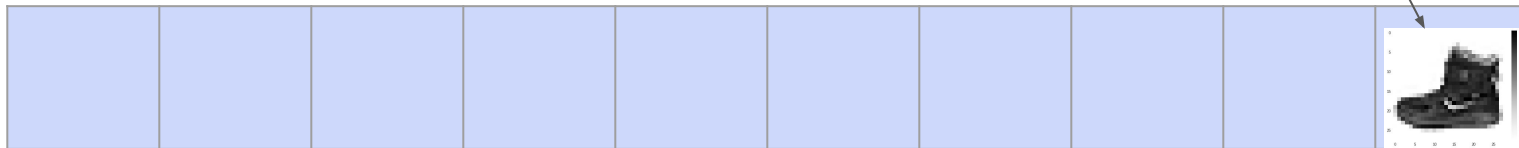



w0

w1

w2

$$w_0x_0 + w_1x_1 + w_2x_2 \dots w_Nx_N = 9$$



11,067 views

43 0 SHARE SAVE ...

■ Why Non-linear Activation Functions

Complete User Registration
system using PHP and MySQL...

Awa Melvine
5.7M views

32:43

```
mnist = tf.keras.datasets.fashion_mnist
(training_images, training_labels), (test_images, test_labels) = mnist.load_data()
training_images=training_images/255.0
test_images=test_images/255.0
model = tf.keras.models.Sequential([
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512, activation=tf.nn.relu),
    tf.keras.layers.Dense(10, activation=tf.nn.softmax)
])
model.compile(optimizer=tf.optimizers.Adam(), loss='sparse_categorical_crossentropy')
model.fit(training_images, training_labels, epochs=5)
```

```
mnist = tf.keras.datasets.fashion_mnist
(training_images, training_labels), (test_images, test_labels) = mnist.load_data()
training_images=training_images/255.0
test_images=test_images/255.0
model = tf.keras.models.Sequential([
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512, activation=tf.nn.relu),
    tf.keras.layers.Dense(10, activation=tf.nn.softmax)
])
model.compile(optimizer=tf.optimizers.Adam(), loss='sparse_categorical_crossentropy')
model.fit(training_images, training_labels, epochs=5)
```

```
class myCallback(tf.keras.callbacks.Callback):  
    def on_epoch_end(self, epoch, logs={}):  
        if(logs.get('loss')<0.4):  
            print("\nLoss is low so cancelling training!")  
            self.model.stop_training = True
```

```
class myCallback(tf.keras.callbacks.Callback):  
    def on_epoch_end(self, epoch, logs={}):  
        if(logs.get('loss')<0.4):  
            print("\nLoss is low so cancelling training!")  
            self.model.stop_training = True
```

```
class myCallback(tf.keras.callbacks.Callback):  
    def on_epoch_end(self, epoch, logs={}):  
        if(logs.get('loss')<0.4):  
            print("\nLoss is low so cancelling training!")  
            self.model.stop_training = True
```

```
mnist = tf.keras.datasets.fashion_mnist
(training_images, training_labels), (test_images, test_labels) = mnist.load_data()
training_images=training_images/255.0
test_images=test_images/255.0
model = tf.keras.models.Sequential([
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512, activation=tf.nn.relu),
    tf.keras.layers.Dense(10, activation=tf.nn.softmax)
])
model.compile(optimizer=tf.optimizers.Adam(), loss='sparse_categorical_crossentropy')
model.fit(training_images, training_labels, epochs=5)
```



```
callbacks = myCallback()
```

```
mnist = tf.keras.datasets.fashion_mnist
(training_images, training_labels), (test_images, test_labels) = mnist.load_data()
training_images=training_images/255.0
test_images=test_images/255.0
model = tf.keras.models.Sequential([
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512, activation=tf.nn.relu),
    tf.keras.layers.Dense(10, activation=tf.nn.softmax)
])
model.compile(optimizer=tf.optimizers.Adam(), loss='sparse_categorical_crossentropy')
model.fit(training_images, training_labels, epochs=5)
```

```
callbacks = myCallback()

mnist = tf.keras.datasets.fashion_mnist
(training_images, training_labels), (test_images, test_labels) = mnist.load_data()
training_images=training_images/255.0
test_images=test_images/255.0
model = tf.keras.models.Sequential([
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512, activation=tf.nn.relu),
    tf.keras.layers.Dense(10, activation=tf.nn.softmax)
])
model.compile(optimizer=tf.optimizers.Adam(), loss='sparse_categorical_crossentropy')
model.fit(training_images, training_labels, epochs=5, callbacks=[callbacks])
```