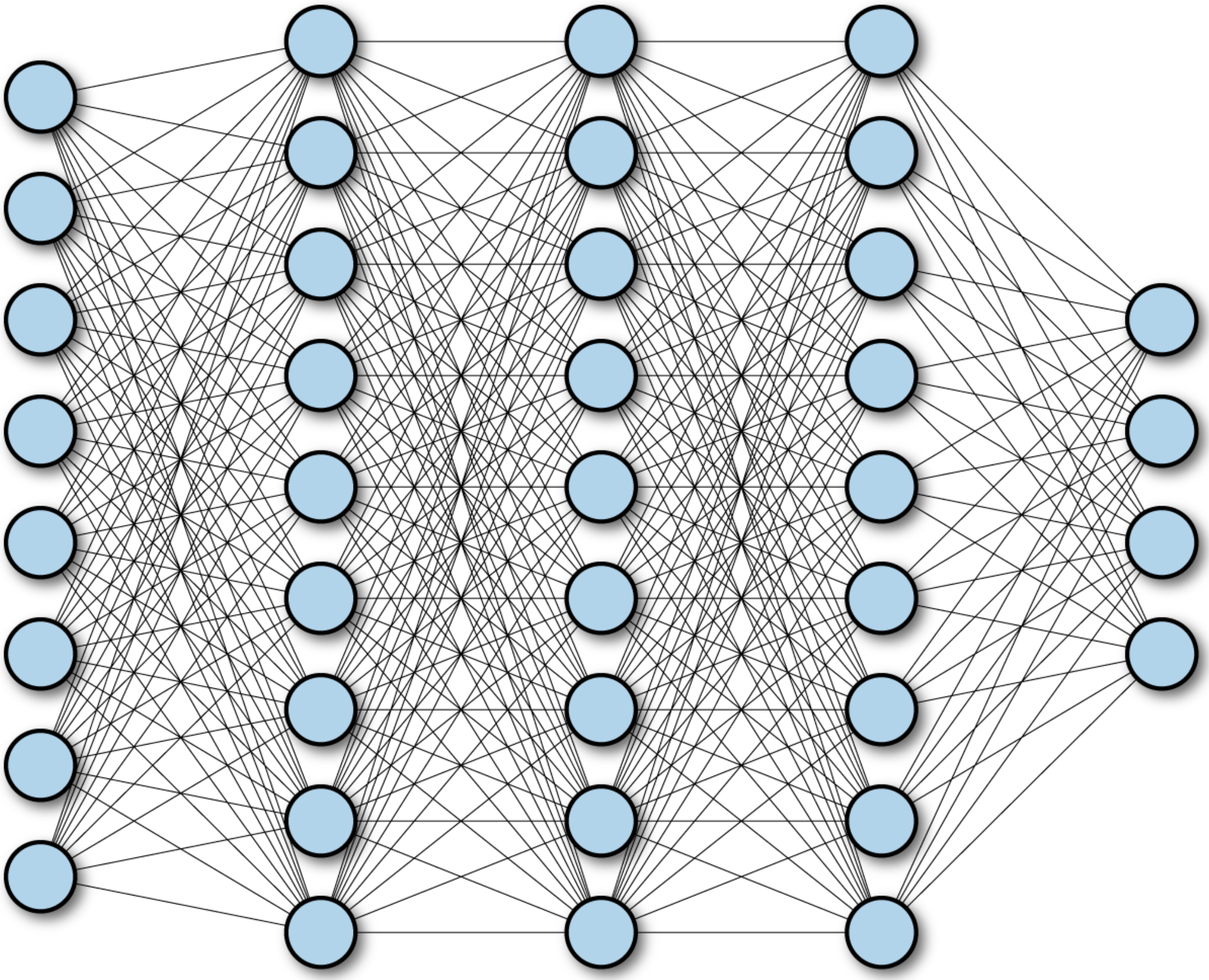
Fully connected example



3 ways to create a Machine Learning model with Keras and TensorFlow 2.0 (Sequential, Functional, and Model Subclassing)

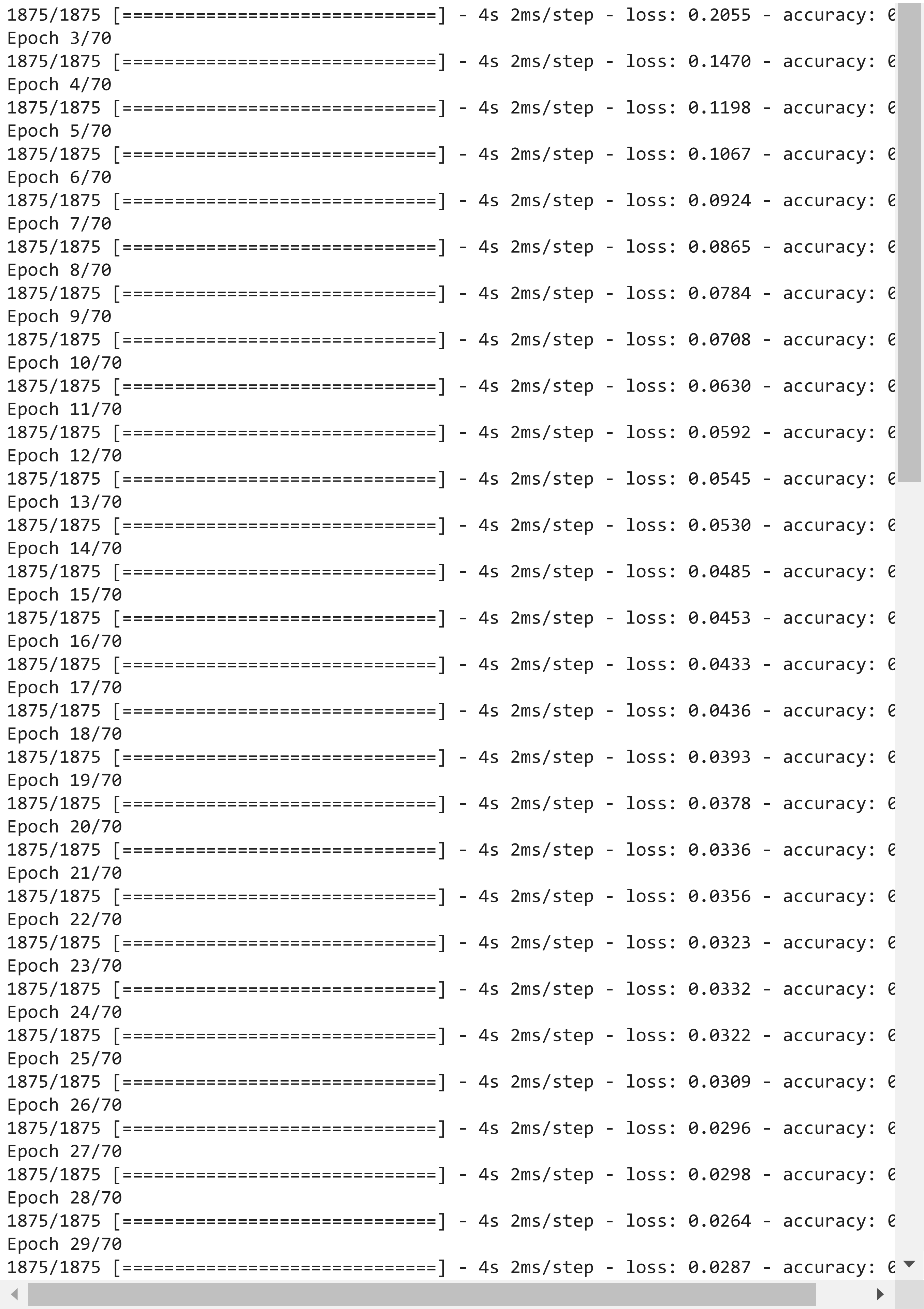
1. Sequential Model is the easiest way to get up and running with Keras in TensorFlow 2.0
2. Functional API is for more complex models, in particular model with multiple inputs or outputs.
3. Model Subclassing is fully-customizable and enables us to implement our own custom forward-pass of the model

```
!pip install mnist
```

```
Collecting mnist
  Downloading https://files.pythonhosted.org/packages/c6/c4/5db3bfe009f8d71f1d532bbadbd
Requirement already satisfied: numpy in /usr/local/lib/python3.6/dist-packages (from mn
Installing collected packages: mnist
Successfully installed mnist-0.2.2
```

```
import numpy as np
import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.utils import to_categorical

train_images = mnist.train_images()
train_labels = mnist.train_labels()
test_images = mnist.test_images()
test_labels = mnist.test_labels()

# Normalize the images.
train_images = (train_images / 255) - 0.5
test_images = (test_images / 255) - 0.5

# Flatten the images.
train_images = train_images.reshape((-1, 784))
test_images = test_images.reshape((-1, 784))

# Build the model.
model = Sequential([
  Dense(64, activation='relu', input_shape=(784,)),
  Dense(64, activation='relu'),
  Dense(10, activation='softmax'),
])

# Compile the model.
model.compile(
  optimizer='adam',
  loss='categorical_crossentropy',
  metrics=['accuracy'],
)
# Train the model.
model.fit(
  train_images,
  to_categorical(train_labels),
  epochs=70,
  batch_size=32,
)
```

```
Epoch 1/70
1875/1875 [==============================] - 6s 2ms/step - loss: 0.5792 - accuracy:
Epoch 2/70
```



```
1875/1875 [==============================] - 4s 2ms/step - loss: 0.2055 - accuracy: 0
Epoch 3/70
1875/1875 [==============================] - 4s 2ms/step - loss: 0.1470 - accuracy: 0
Epoch 4/70
1875/1875 [==============================] - 4s 2ms/step - loss: 0.1198 - accuracy: 0
Epoch 5/70
1875/1875 [==============================] - 4s 2ms/step - loss: 0.1067 - accuracy: 0
Epoch 6/70
1875/1875 [==============================] - 4s 2ms/step - loss: 0.0924 - accuracy: 0
Epoch 7/70
1875/1875 [==============================] - 4s 2ms/step - loss: 0.0865 - accuracy: 0
Epoch 8/70
1875/1875 [==============================] - 4s 2ms/step - loss: 0.0784 - accuracy: 0
Epoch 9/70
1875/1875 [==============================] - 4s 2ms/step - loss: 0.0708 - accuracy: 0
Epoch 10/70
1875/1875 [==============================] - 4s 2ms/step - loss: 0.0630 - accuracy: 0
Epoch 11/70
1875/1875 [==============================] - 4s 2ms/step - loss: 0.0592 - accuracy: 0
Epoch 12/70
1875/1875 [==============================] - 4s 2ms/step - loss: 0.0545 - accuracy: 0
Epoch 13/70
1875/1875 [==============================] - 4s 2ms/step - loss: 0.0530 - accuracy: 0
Epoch 14/70
1875/1875 [==============================] - 4s 2ms/step - loss: 0.0485 - accuracy: 0
Epoch 15/70
1875/1875 [==============================] - 4s 2ms/step - loss: 0.0453 - accuracy: 0
Epoch 16/70
1875/1875 [==============================] - 4s 2ms/step - loss: 0.0433 - accuracy: 0
Epoch 17/70
1875/1875 [==============================] - 4s 2ms/step - loss: 0.0436 - accuracy: 0
Epoch 18/70
1875/1875 [==============================] - 4s 2ms/step - loss: 0.0393 - accuracy: 0
Epoch 19/70
1875/1875 [==============================] - 4s 2ms/step - loss: 0.0378 - accuracy: 0
Epoch 20/70
1875/1875 [==============================] - 4s 2ms/step - loss: 0.0336 - accuracy: 0
Epoch 21/70
1875/1875 [==============================] - 4s 2ms/step - loss: 0.0356 - accuracy: 0
Epoch 22/70
1875/1875 [==============================] - 4s 2ms/step - loss: 0.0323 - accuracy: 0
Epoch 23/70
1875/1875 [==============================] - 4s 2ms/step - loss: 0.0332 - accuracy: 0
Epoch 24/70
1875/1875 [==============================] - 4s 2ms/step - loss: 0.0322 - accuracy: 0
Epoch 25/70
1875/1875 [==============================] - 4s 2ms/step - loss: 0.0309 - accuracy: 0
Epoch 26/70
1875/1875 [==============================] - 4s 2ms/step - loss: 0.0296 - accuracy: 0
Epoch 27/70
1875/1875 [==============================] - 4s 2ms/step - loss: 0.0298 - accuracy: 0
Epoch 28/70
1875/1875 [==============================] - 4s 2ms/step - loss: 0.0264 - accuracy: 0
Epoch 29/70
1875/1875 [==============================] - 4s 2ms/step - loss: 0.0287 - accuracy: 0
```

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

```
313/313 [==============================] - 1s 2ms/step - loss: 0.2023 - accuracy: 0.972
[0.20229823887348175, 0.972000002861023]
```

```
model.save_weights('model.h5')
```

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

# Build the model.
model = Sequential([
  Dense(64, activation='relu', input_shape=(784,)),
  Dense(64, activation='relu'),
  Dense(10, activation='softmax'),
])

# Load the model's saved weights.
model.load_weights('model.h5')
```

```
# Predict on the first 5 test images.
predictions = model.predict(test_images[:20])

# Print our model's predictions.
print(np.argmax(predictions, axis=1)) # [7, 2, 1, 0, 4]

# Check our predictions against the ground truths.
print(test_labels[:20]) # [7, 2, 1, 0, 4]
```

```
[7 2 1 0 4 1 4 9 5 9 0 6 9 0 1 5 9 7 8 4]
[7 2 1 0 4 1 4 9 5 9 0 6 9 0 1 5 9 7 3 4]
```

```
import numpy as np
import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.utils import to_categorical

train_images = mnist.train_images()
train_labels = mnist.train_labels()
test_images = mnist.test_images()
test_labels = mnist.test_labels()

# Normalize the images.
train_images = (train_images / 255) - 0.5
train_images = (train_images / 255) - 0.5
test_images = (test_images / 255) - 0.5

# Flatten the images.
train_images = train_images.reshape((-1, 784))
test_images = test_images.reshape((-1, 784))

# Build the model.
model = Sequential([
  Dense(64, activation='relu', input_shape=(784,)),
  Dense(64, activation='relu'),#sigmoid
  # Dropout(0.5),
  Dense(64, activation='relu'),#sigmoid
  # Dropout(0.5),
  Dense(64, activation='relu'),#sigmoid
  Dense(10, activation='softmax'),
])

# Compile the model.
model.compile(
  optimizer='adam',
  loss='categorical_crossentropy',
  metrics=['accuracy'],
)

# Train the model.
model.fit(
  train_images,
  to_categorical(train_labels),
  epochs=5,
  batch_size=32,
  validation_data=(test_images, to_categorical(test_labels))

)
```

```
Epoch 1/5
1875/1875 [==============================] - 5s 3ms/step - loss: 0.6071 - accuracy: 0.8
Epoch 2/5
1875/1875 [==============================] - 5s 2ms/step - loss: 0.1870 - accuracy: 0.9
Epoch 3/5
1875/1875 [==============================] - 5s 2ms/step - loss: 0.1459 - accuracy: 0.9
Epoch 4/5
1875/1875 [==============================] - 5s 2ms/step - loss: 0.1293 - accuracy: 0.9
Epoch 5/5
1875/1875 [==============================] - 5s 3ms/step - loss: 0.1120 - accuracy: 0.9
<tensorflow.python.keras.callbacks.History at 0x7f53807495c0>
```