```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.optimizers import RMSprop
from tensorflow.keras.callbacks import EarlyStopping
import matplotlib.pyplot as plt
from tensorflow.keras.datasets import mnist
from tensorflow.keras.utils import to_categorical


(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train = x_train.reshape(60000, 784).astype("float32") / 255
x_test = x_test.reshape(10000, 784).astype("float32") / 255
y_train = to_categorical(y_train, 10)
y_test = to_categorical(y_test, 10)
model_4 = Sequential()
model_4.add(Dense(400, activation="relu", input_shape=(784,)))
model_4.add(Dropout(0.4))
model_4.add(Dense(300, activation="relu"))
model_4.add(Dropout(0.4))
model_4.add(Dense(10, activation="softmax"))
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an `input_shape`/`input_dim` arg
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
learning_rate = 0.003
model_4.compile(
    loss="categorical_crossentropy",
    optimizer=RMSprop(learning_rate=learning_rate),
    metrics=["accuracy"],
)


batch_size = 128
epochs = 20

history_4 = model_4.fit(
    x_train,
    y_train,
    batch_size=batch_size,
    epochs=epochs,
    verbose=1,
    validation_data=(x_test, y_test),
)
```

```
Epoch 1/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 8s 14ms/step - accuracy: 0.8302 - loss: 0.5392 - val_accuracy: 0.9635 - val_loss: 0.1245
Epoch 2/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 9s 11ms/step - accuracy: 0.9505 - loss: 0.1723 - val_accuracy: 0.9684 - val_loss: 0.1111
Epoch 3/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 11s 12ms/step - accuracy: 0.9595 - loss: 0.1414 - val_accuracy: 0.9746 - val_loss: 0.1005
Epoch 4/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 6s 13ms/step - accuracy: 0.9628 - loss: 0.1318 - val_accuracy: 0.9752 - val_loss: 0.0894
Epoch 5/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 9s 12ms/step - accuracy: 0.9692 - loss: 0.1179 - val_accuracy: 0.9763 - val_loss: 0.0955
Epoch 6/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 10s 12ms/step - accuracy: 0.9694 - loss: 0.1130 - val_accuracy: 0.9777 - val_loss: 0.0952
Epoch 7/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 6s 13ms/step - accuracy: 0.9717 - loss: 0.1079 - val_accuracy: 0.9784 - val_loss: 0.0934
Epoch 8/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 5s 11ms/step - accuracy: 0.9744 - loss: 0.0959 - val_accuracy: 0.9772 - val_loss: 0.0961
Epoch 9/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 6s 14ms/step - accuracy: 0.9767 - loss: 0.0911 - val_accuracy: 0.9814 - val_loss: 0.0863
Epoch 10/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 6s 12ms/step - accuracy: 0.9757 - loss: 0.0943 - val_accuracy: 0.9789 - val_loss: 0.1011
Epoch 11/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 10s 12ms/step - accuracy: 0.9755 - loss: 0.0958 - val_accuracy: 0.9793 - val_loss: 0.1007
Epoch 12/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 11s 14ms/step - accuracy: 0.9788 - loss: 0.0882 - val_accuracy: 0.9797 - val_loss: 0.1047
Epoch 13/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 10s 13ms/step - accuracy: 0.9773 - loss: 0.0910 - val_accuracy: 0.9811 - val_loss: 0.0969
Epoch 14/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 6s 12ms/step - accuracy: 0.9786 - loss: 0.0845 - val_accuracy: 0.9800 - val_loss: 0.1016
Epoch 15/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 10s 12ms/step - accuracy: 0.9796 - loss: 0.0877 - val_accuracy: 0.9831 - val_loss: 0.0930
Epoch 16/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 11s 13ms/step - accuracy: 0.9800 - loss: 0.0875 - val_accuracy: 0.9820 - val_loss: 0.0956
Epoch 17/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 11s 14ms/step - accuracy: 0.9823 - loss: 0.0782 - val_accuracy: 0.9805 - val_loss: 0.1092
Epoch 18/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 10s 13ms/step - accuracy: 0.9820 - loss: 0.0772 - val_accuracy: 0.9805 - val_loss: 0.1067
Epoch 19/20
```

```
469/469 ━━━━━━━━━━━━━━━━━━━━ 10s 12ms/step - accuracy: 0.9807 - loss: 0.0826 - val_accuracy: 0.9827 - val_loss: 0.0954
Epoch 20/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 7s 14ms/step - accuracy: 0.9814 - loss: 0.0824 - val_accuracy: 0.9808 - val_loss: 0.1200
```

```
model_4.summary()
score = model_4.evaluate(x_test, y_test, verbose=0)
print("Model 4 - Test loss:", score[0])
print("Model 4 - Test accuracy:", score[1])
```

Model: "sequential"

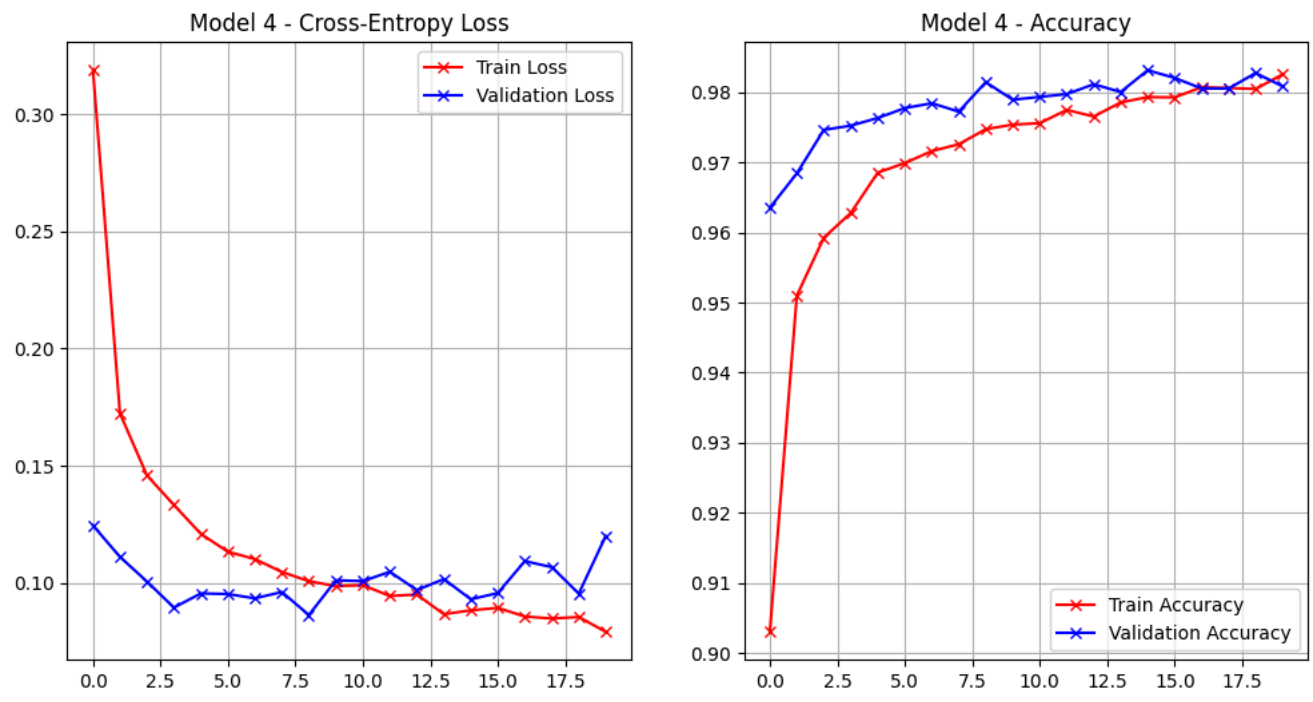
Layer (type)	Output Shape	Param #
dense (Dense)	(None, 400)	314,000
dropout (Dropout)	(None, 400)	0
dense_1 (Dense)	(None, 300)	120,300
dropout_1 (Dropout)	(None, 300)	0
dense_2 (Dense)	(None, 10)	3,010

Total params: 874,622 (3.34 MB)
Trainable params: 437,310 (1.67 MB)
Non-trainable params: 0 (0.00 B)
Optimizer params: 437,312 (1.67 MB)

```
Model 4 - Test loss: 0.1200367659330368
Model 4 - Test accuracy: 0.9807999730110168
```

```
def plot_loss_accuracy(history, model_name="Model"):
    fig = plt.figure(figsize=(12, 6))
    ax = fig.add_subplot(1, 2, 1)
    ax.plot(history.history["loss"], "r-x", label="Train Loss")
    ax.plot(history.history["val_loss"], "b-x", label="Validation Loss")
    ax.legend()
    ax.set_title(f"{model_name} - Cross-Entropy Loss")
    ax.grid(True)
    ax = fig.add_subplot(1, 2, 2)
    acc_key = "accuracy" if "accuracy" in history.history else "acc"
    val_acc_key = "val_accuracy" if "val_accuracy" in history.history else "val_acc"

    ax.plot(history.history[acc_key], "r-x", label="Train Accuracy")
    ax.plot(history.history[val_acc_key], "b-x", label="Validation Accuracy")
    ax.legend()
    ax.set_title(f"{model_name} - Accuracy")
    ax.grid(True)
plot_loss_accuracy(history_4, model_name="Model 4")
```



Start coding or generate with AI.