In [106…

```
pd.read_csv('model_trains5.csv')
```

Out[106…

	Test #	Number of Layers	Nodes per Layer	Activation	Processing Time	Training Set Accuracy	Test Set Accuracy
0	1	3	300,100,10	relu, relu, softmax	02:03.1	0.9994	0.97264
1	2	2	300, 100	relu, softmax	01:41.7	0.9732	0.96942
2	3	5	100, 10	4 relu then softmax	01:42.4	0.9740	0.96953
3	4	6	100,10, 10	4 relu then 2 softmax	02:20.5	0.9740	0.96953
4	5	4	900,300,100,10	3 relu then 1 softmax	07:58.4	0.9786	0.97385

It is very awkard to parse PDFs and I keep getting errors so I am providing my paper in the markdown code.

(1) Data preparation, exploration, visualization:

This data preperation was easy as the MNST dataset is built for this type of analysis and input.

(2) Review research design and modeling methods:

I really found that relu followed by softmax activation works well.

(3) Review results, evaluate models:

My final model took awile to train but I really believe that it works really well based on the test results. Some images in the dataset are clearly illegible so I dont want the model to accurately predict those.

(4) Implementation and programming:

Programming is pretty straight forward, I had a few issues with Tensorflow in the begginning but I eventually worked oyut all the kinks.

(5) Exposition, problem description, and management recommendations:

Based on the benchmark study above I would state that the most trustworthy parameters would be test 5 with 3 relu activated layers and 900, 300, and 100 nodes per layer respectively. Additionally a softmax layer with 10 activation nodes for each of the digits after a large hidden network works best when predicting this dataset. Rather then adding in other methods of activation this combination of Rectified Linear Unit activation feeding into a final softmax activation seems to work vest for categorical output. Additionally if you look at the model predictions below the model was only incorrect on very poorly drawn numbers and thus will work very well for individuals with handwriting within the acceptable norm.

In [45]:

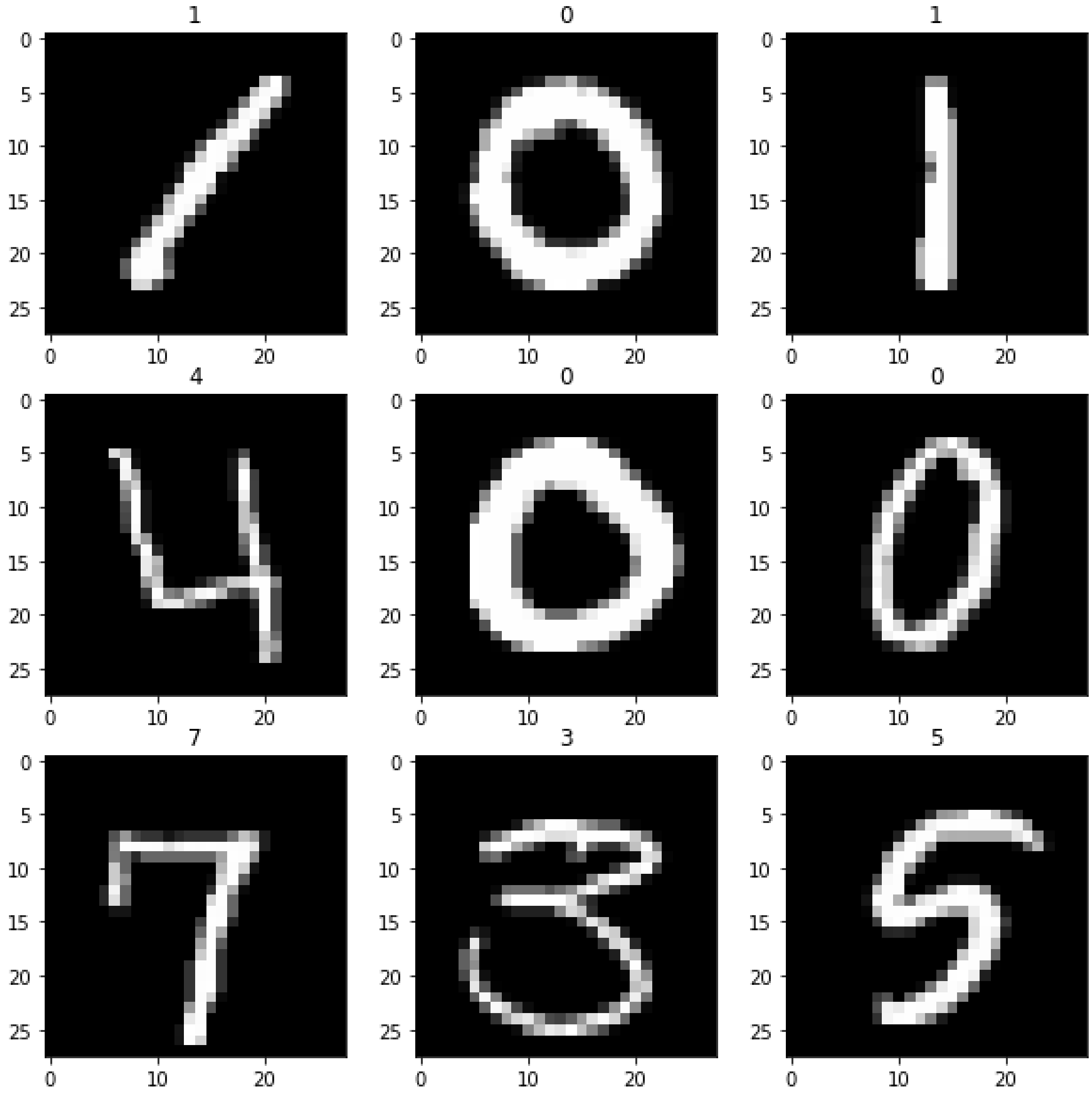
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
test_df = pd.read_csv('test.csv')
sample_sub = pd.read_csv('sample_submission.csv')
train_df = pd.read_csv('train.csv')
y=train_df['label'].to_numpy()
X=train_df.loc[:, train_df.columns != 'label'].to_numpy()
X_test=test_df.to_numpy()
X
```

Out[45]:

```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]], dtype=int64)
```

In [46]:

```
X_train_full = X.reshape(X.shape[0], 28, 28)
plt.figure(figsize = (10,10))
for i in range(0, 9):
    plt.subplot(330 + (i+1))
    plt.imshow(X_train_full[i], cmap=plt.get_cmap('gray'))
    plt.title(y[i])
```



In [47]:

```
import tensorflow as tf
from tensorflow import keras
model = keras.models.Sequential()
model.add(keras.layers.Flatten(input_shape=[28, 28]))
model.add(keras.layers.Dense(300, activation="relu"))
model.add(keras.layers.Dense(100, activation="relu"))
model.add(keras.layers.Dense(10, activation="softmax"))
keras.backend.clear_session()
np.random.seed(42)
tf.random.set_seed(42)
model = keras.models.Sequential([
    keras.layers.Flatten(input_shape=[28, 28]),
    keras.layers.Dense(300, activation="relu"),
    keras.layers.Dense(100, activation="relu"),
    keras.layers.Dense(10, activation="softmax")
])
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
flatten (Flatten)            (None, 784)               0         
_________________________________________________________________
dense (Dense)                (None, 300)               235500    
_________________________________________________________________
```

```
dense_1 (Dense)              (None, 100)               30100     
_________________________________________________________________
dense_2 (Dense)              (None, 10)                1010      
=================================================================
Total params: 266,610
Trainable params: 266,610
Non-trainable params: 0
_________________________________________________________________
```

In [48]:

```
model.compile(loss="sparse_categorical_crossentropy",
              optimizer="sgd",
              metrics=["accuracy"])
```

In [49]:

```
X_train_full=X_train_full.astype(int)
y_train_full=y_train_full.astype(int)
X_test=X_test.astype(int)
```

In [50]:

```
X_valid, X_train = X_train_full[:5000] / 255, X_train_full[5000:] / 255
y_valid, y_train = y[:5000], y[5000:]
X_test = X_test / 255
```

In [51]:

```
import datetime
from datetime import *
start=datetime.now()
history = model.fit(X_train, y_train, epochs=60, validation_data=(X_valid, y_valid))
end=datetime.now()
print(end-start)
```

```
Epoch 1/60
1157/1157 [==============================] - 3s 2ms/step - loss: 0.7392 - accuracy: 0.8126 - val_l
oss: 0.3770 - val_accuracy: 0.8942
Epoch 2/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.3273 - accuracy: 0.9067 - val_l
oss: 0.2907 - val_accuracy: 0.9190
Epoch 3/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.2724 - accuracy: 0.9219 - val_l
oss: 0.2553 - val_accuracy: 0.9296
Epoch 4/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.2388 - accuracy: 0.9309 - val_l
oss: 0.2382 - val_accuracy: 0.9346
Epoch 5/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.2120 - accuracy: 0.9388 - val_l
oss: 0.2058 - val_accuracy: 0.9424
Epoch 6/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.1902 - accuracy: 0.9456 - val_l
oss: 0.1897 - val_accuracy: 0.9460
Epoch 7/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.1727 - accuracy: 0.9512 - val_l
oss: 0.1770 - val_accuracy: 0.9498
Epoch 8/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.1568 - accuracy: 0.9559 - val_l
oss: 0.1646 - val_accuracy: 0.9540
Epoch 9/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.1443 - accuracy: 0.9587 - val_l
oss: 0.1620 - val_accuracy: 0.9502
Epoch 10/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.1330 - accuracy: 0.9624 - val_l
oss: 0.1438 - val_accuracy: 0.9586
Epoch 11/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.1234 - accuracy: 0.9648 - val_l
oss: 0.1376 - val_accuracy: 0.9592
Epoch 12/60
```

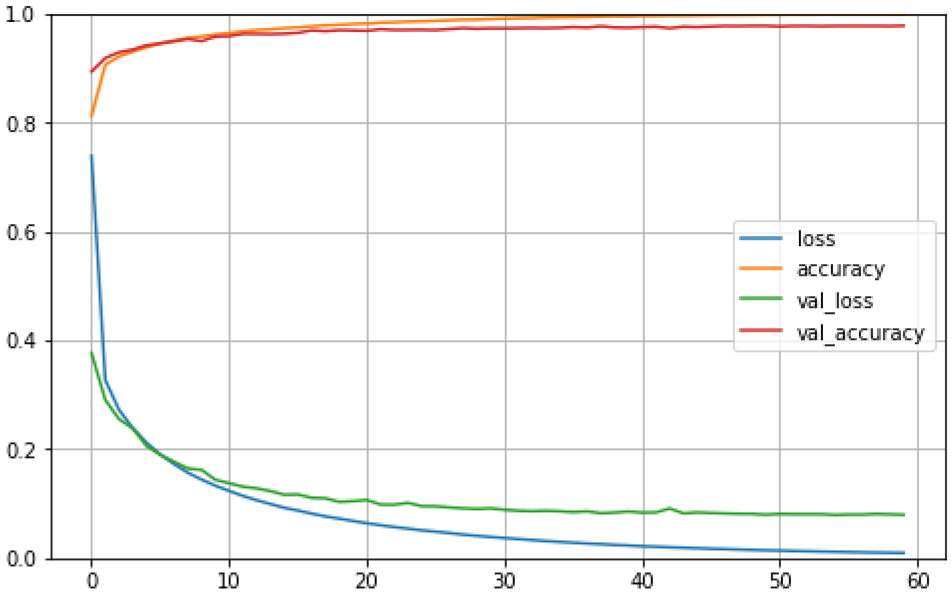
```
1157/1157 [==============================] - 2s 2ms/step - loss: 0.1142 - accuracy: 0.9682 - val_l
oss: 0.1311 - val_accuracy: 0.9636
Epoch 13/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.1063 - accuracy: 0.9704 - val_l
oss: 0.1282 - val_accuracy: 0.9634
Epoch 14/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0993 - accuracy: 0.9719 - val_l
oss: 0.1231 - val_accuracy: 0.9630
Epoch 15/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0925 - accuracy: 0.9740 - val_l
oss: 0.1164 - val_accuracy: 0.9638
Epoch 16/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0873 - accuracy: 0.9753 - val_l
oss: 0.1169 - val_accuracy: 0.9654
Epoch 17/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0817 - accuracy: 0.9774 - val_l
oss: 0.1107 - val_accuracy: 0.9700
Epoch 18/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0765 - accuracy: 0.9791 - val_l
oss: 0.1098 - val_accuracy: 0.9684
Epoch 19/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0724 - accuracy: 0.9802 - val_l
oss: 0.1033 - val_accuracy: 0.9704
Epoch 20/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0681 - accuracy: 0.9815 - val_l
oss: 0.1046 - val_accuracy: 0.9698
Epoch 21/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0641 - accuracy: 0.9825 - val_l
oss: 0.1067 - val_accuracy: 0.9692
Epoch 22/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0604 - accuracy: 0.9841 - val_l
oss: 0.0984 - val_accuracy: 0.9720
Epoch 23/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0570 - accuracy: 0.9849 - val_l
oss: 0.0982 - val_accuracy: 0.9706
Epoch 24/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0541 - accuracy: 0.9860 - val_l
oss: 0.1014 - val_accuracy: 0.9708
Epoch 25/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0510 - accuracy: 0.9865 - val_l
oss: 0.0954 - val_accuracy: 0.9710
Epoch 26/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0485 - accuracy: 0.9874 - val_l
oss: 0.0951 - val_accuracy: 0.9704
Epoch 27/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0460 - accuracy: 0.9885 - val_l
oss: 0.0931 - val_accuracy: 0.9724
Epoch 28/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0431 - accuracy: 0.9894 - val_l
oss: 0.0913 - val_accuracy: 0.9740
Epoch 29/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0408 - accuracy: 0.9897 - val_l
oss: 0.0904 - val_accuracy: 0.9730
Epoch 30/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0388 - accuracy: 0.9908 - val_l
oss: 0.0913 - val_accuracy: 0.9738
Epoch 31/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0368 - accuracy: 0.9915 - val_l
oss: 0.0888 - val_accuracy: 0.9736
Epoch 32/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0350 - accuracy: 0.9923 - val_l
oss: 0.0869 - val_accuracy: 0.9740
Epoch 33/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0332 - accuracy: 0.9926 - val_l
oss: 0.0862 - val_accuracy: 0.9744
Epoch 34/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0313 - accuracy: 0.9935 - val_l
oss: 0.0869 - val_accuracy: 0.9740
Epoch 35/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0298 - accuracy: 0.9940 - val_l
```

```
oss: 0.0860 - val_accuracy: 0.9746
Epoch 36/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0283 - accuracy: 0.9943 - val_l
oss: 0.0846 - val_accuracy: 0.9760
Epoch 37/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0268 - accuracy: 0.9946 - val_l
oss: 0.0859 - val_accuracy: 0.9750
Epoch 38/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0256 - accuracy: 0.9954 - val_l
oss: 0.0825 - val_accuracy: 0.9776
Epoch 39/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0242 - accuracy: 0.9959 - val_l
oss: 0.0836 - val_accuracy: 0.9752
Epoch 40/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0232 - accuracy: 0.9959 - val_l
oss: 0.0856 - val_accuracy: 0.9748
Epoch 41/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0218 - accuracy: 0.9965 - val_l
oss: 0.0834 - val_accuracy: 0.9758
Epoch 42/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0208 - accuracy: 0.9966 - val_l
oss: 0.0838 - val_accuracy: 0.9768
Epoch 43/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0198 - accuracy: 0.9969 - val_l
oss: 0.0911 - val_accuracy: 0.9738
Epoch 44/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0189 - accuracy: 0.9970 - val_l
oss: 0.0822 - val_accuracy: 0.9768
Epoch 45/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0181 - accuracy: 0.9974 - val_l
oss: 0.0839 - val_accuracy: 0.9760
Epoch 46/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0171 - accuracy: 0.9976 - val_l
oss: 0.0828 - val_accuracy: 0.9770
Epoch 47/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0165 - accuracy: 0.9979 - val_l
oss: 0.0820 - val_accuracy: 0.9780
Epoch 48/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0157 - accuracy: 0.9981 - val_l
oss: 0.0814 - val_accuracy: 0.9778
Epoch 49/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0149 - accuracy: 0.9983 - val_l
oss: 0.0812 - val_accuracy: 0.9782
Epoch 50/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0142 - accuracy: 0.9986 - val_l
oss: 0.0798 - val_accuracy: 0.9782
Epoch 51/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0137 - accuracy: 0.9985 - val_l
oss: 0.0811 - val_accuracy: 0.9772
Epoch 52/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0130 - accuracy: 0.9988 - val_l
oss: 0.0807 - val_accuracy: 0.9780
Epoch 53/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0125 - accuracy: 0.9988 - val_l
oss: 0.0806 - val_accuracy: 0.9780
Epoch 54/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0119 - accuracy: 0.9990 - val_l
oss: 0.0806 - val_accuracy: 0.9776
Epoch 55/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0115 - accuracy: 0.9992 - val_l
oss: 0.0793 - val_accuracy: 0.9780
Epoch 56/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0109 - accuracy: 0.9991 - val_l
oss: 0.0799 - val_accuracy: 0.9780
Epoch 57/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0105 - accuracy: 0.9992 - val_l
oss: 0.0799 - val_accuracy: 0.9780
Epoch 58/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0101 - accuracy: 0.9994 - val_l
oss: 0.0811 - val_accuracy: 0.9780
```

```
Epoch 59/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0097 - accuracy: 0.9995 - val_l
oss: 0.0804 - val_accuracy: 0.9778
Epoch 60/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0094 - accuracy: 0.9994 - val_l
oss: 0.0795 - val_accuracy: 0.9784
0:02:03.088101
```

In [52]:

```
pd.DataFrame(history.history).plot(figsize=(8, 5))
plt.grid(True)
plt.gca().set_ylim(0, 1)
plt.show()
```



In [53]:

```
y_pred=model.predict_classes(X_test)
y_pred
```

```
WARNING:tensorflow:Model was constructed with shape (None, 28, 28) for input KerasTensor(type_spec
=TensorSpec(shape=(None, 28, 28), dtype=tf.float32, name='flatten_input'), name='flatten_input', d
escription="created by layer 'flatten_input'"), but it was called on an input with incompatible sh
ape (32, 784).
```

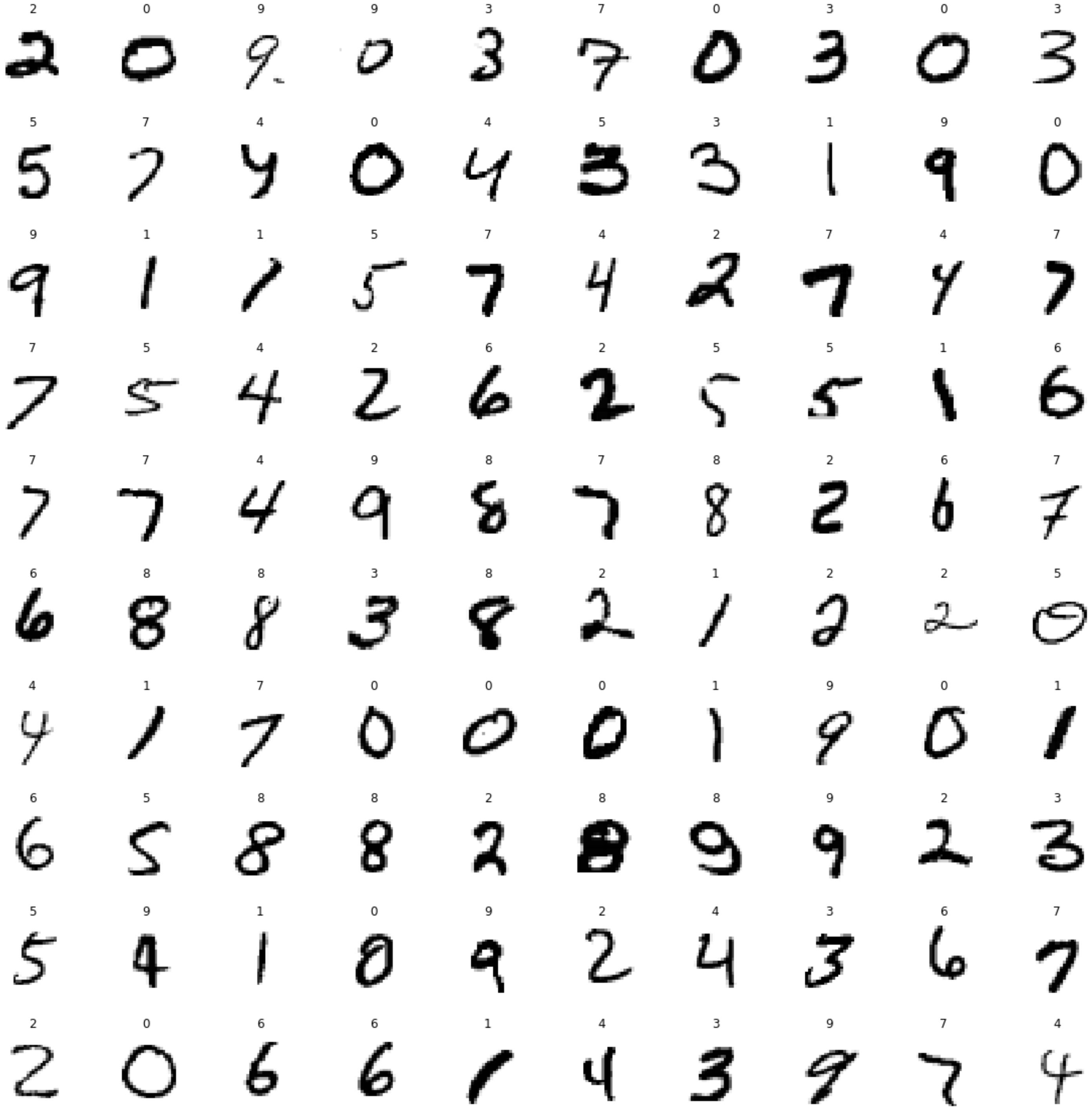
```
C:\Users\rocchm1\AppData\Roaming\Python\Python37\site-packages\tensorflow\python\keras\engine\sequ
ential.py:455: UserWarning: `model.predict_classes()` is deprecated and will be removed after 2021
-01-01. Please use instead:* `np.argmax(model.predict(x), axis=-1)`,   if your model does multi-cl
ass classification   (e.g. if it uses a `softmax` last-layer activation).* `(model.predict(x) > 0.
5).astype("int32")`,   if your model does binary classification   (e.g. if it uses a `sigmoid` las
t-layer activation).
  warnings.warn('`model.predict_classes()` is deprecated and '
```

Out[53]:

```
array([2, 0, 9, ..., 3, 9, 2], dtype=int64)
```

In [57]:

```
plt.figure(figsize=(20,20))
for index, image in enumerate(X_test[:100].reshape(100, 28, 28)):
    plt.subplot(10, 10, index + 1)
    plt.imshow(image, cmap="binary", interpolation="nearest")
    plt.axis('off')
    plt.title(y_pred[index], fontsize=12)
plt.subplots_adjust(wspace=0.2, hspace=0.5)
plt.show()
```



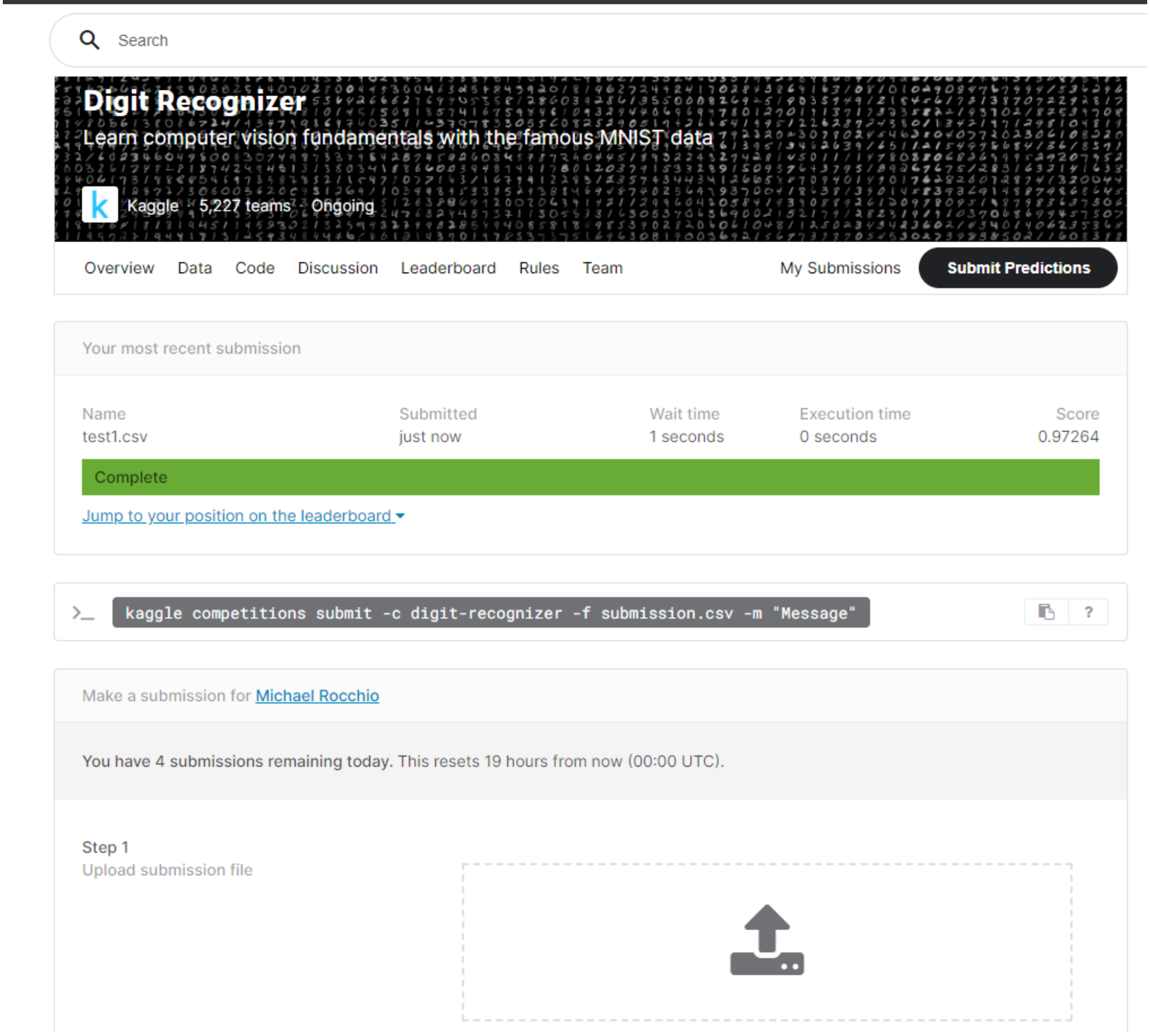
In [58]:

```
sample_sub['Label']=y_pred
sample_sub.to_csv('test1.csv', index=False)
```

In [92]:

```
from IPython.display import Image
Image(filename='test1.png')
######
######  MY USERNAME IN KAGGLE IS michaelrocchio ######
######
```

Out[92]:



Your most recent submission

Name	Submitted	Wait time	Execution time	Score
test1.csv	just now	1 seconds	0 seconds	0.97264

Complete

Jump to your position on the leaderboard

```
kaggle competitions submit -c digit-recognizer -f submission.csv -m "Message"
```

Make a submission for Michael Rocchio

You have 4 submissions remaining today. This resets 19 hours from now (00:00 UTC).

In [59]:

```
model = keras.models.Sequential()
model.add(keras.layers.Flatten(input_shape=[28, 28]))
model.add(keras.layers.Dense(300, activation="relu"))
model.add(keras.layers.Dense(10, activation="softmax"))
keras.backend.clear_session()
np.random.seed(42)
tf.random.set_seed(42)
model = keras.models.Sequential([
    keras.layers.Flatten(input_shape=[28, 28]),
    keras.layers.Dense(300, activation="relu"),
    keras.layers.Dense(10, activation="softmax")
])
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
flatten (Flatten)            (None, 784)               0         
_________________________________________________________________
dense (Dense)                (None, 300)               235500    
_________________________________________________________________
dense_1 (Dense)              (None, 10)                3010      
```

```
=================================================================
Total params: 238,510
Trainable params: 238,510
Non-trainable params: 0
_________________________________________________________________
```

In [60]:

```
model.compile(loss="sparse_categorical_crossentropy",
              optimizer="sgd",
              metrics=["accuracy"])
```

In [61]:

```
start=datetime.now()
history = model.fit(X_train, y_train, epochs=60, validation_data=(X_valid, y_valid))
end=datetime.now()
print(end-start)
```

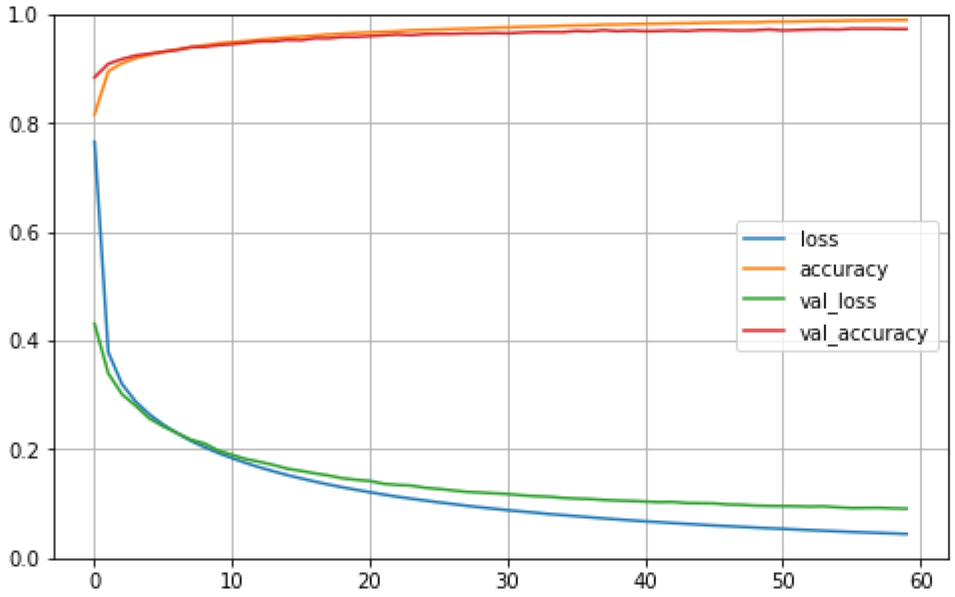
```
Epoch 1/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.7660 - accuracy: 0.8163 - val_l
oss: 0.4310 - val_accuracy: 0.8844
Epoch 2/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.3785 - accuracy: 0.8956 - val_l
oss: 0.3397 - val_accuracy: 0.9092
Epoch 3/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.3200 - accuracy: 0.9102 - val_l
oss: 0.3013 - val_accuracy: 0.9180
Epoch 4/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.2878 - accuracy: 0.9196 - val_l
oss: 0.2798 - val_accuracy: 0.9246
Epoch 5/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.2641 - accuracy: 0.9259 - val_l
oss: 0.2565 - val_accuracy: 0.9280
Epoch 6/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.2454 - accuracy: 0.9315 - val_l
oss: 0.2422 - val_accuracy: 0.9314
Epoch 7/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.2297 - accuracy: 0.9357 - val_l
oss: 0.2301 - val_accuracy: 0.9346
Epoch 8/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.2157 - accuracy: 0.9397 - val_l
oss: 0.2178 - val_accuracy: 0.9398
Epoch 9/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.2039 - accuracy: 0.9435 - val_l
oss: 0.2107 - val_accuracy: 0.9404
Epoch 10/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.1932 - accuracy: 0.9468 - val_l
oss: 0.1976 - val_accuracy: 0.9434
Epoch 11/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.1838 - accuracy: 0.9491 - val_l
oss: 0.1899 - val_accuracy: 0.9454
Epoch 12/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.1749 - accuracy: 0.9512 - val_l
oss: 0.1822 - val_accuracy: 0.9478
Epoch 13/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.1669 - accuracy: 0.9537 - val_l
oss: 0.1771 - val_accuracy: 0.9504
Epoch 14/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.1596 - accuracy: 0.9556 - val_l
oss: 0.1713 - val_accuracy: 0.9506
Epoch 15/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.1527 - accuracy: 0.9578 - val_l
oss: 0.1645 - val_accuracy: 0.9532
Epoch 16/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.1467 - accuracy: 0.9594 - val_l
oss: 0.1604 - val_accuracy: 0.9526
Epoch 17/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.1408 - accuracy: 0.9612 - val_l
oss: 0.1559 - val_accuracy: 0.9562
Epoch 18/60
```

```
1157/1157 [==============================] - 2s 2ms/step - loss: 0.1353 - accuracy: 0.9632 - val_l
oss: 0.1519 - val_accuracy: 0.9562
Epoch 19/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.1303 - accuracy: 0.9648 - val_l
oss: 0.1470 - val_accuracy: 0.9588
Epoch 20/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.1256 - accuracy: 0.9661 - val_l
oss: 0.1440 - val_accuracy: 0.9588
Epoch 21/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.1212 - accuracy: 0.9672 - val_l
oss: 0.1420 - val_accuracy: 0.9602
Epoch 22/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.1170 - accuracy: 0.9683 - val_l
oss: 0.1367 - val_accuracy: 0.9618
Epoch 23/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.1129 - accuracy: 0.9693 - val_l
oss: 0.1347 - val_accuracy: 0.9632
Epoch 24/60
1157/1157 [==============================] - ETA: 0s - loss: 0.1094 - accuracy: 0.97 - 2s 2ms/step
- loss: 0.1093 - accuracy: 0.9709 - val_loss: 0.1334 - val_accuracy: 0.9620
Epoch 25/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.1060 - accuracy: 0.9716 - val_l
oss: 0.1294 - val_accuracy: 0.9640
Epoch 26/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.1026 - accuracy: 0.9722 - val_l
oss: 0.1273 - val_accuracy: 0.9644
Epoch 27/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.0996 - accuracy: 0.9733 - val_l
oss: 0.1246 - val_accuracy: 0.9644
Epoch 28/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0963 - accuracy: 0.9740 - val_l
oss: 0.1219 - val_accuracy: 0.9654
Epoch 29/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.0935 - accuracy: 0.9748 - val_l
oss: 0.1207 - val_accuracy: 0.9652
Epoch 30/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0909 - accuracy: 0.9758 - val_l
oss: 0.1192 - val_accuracy: 0.9660
Epoch 31/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.0882 - accuracy: 0.9763 - val_l
oss: 0.1179 - val_accuracy: 0.9654
Epoch 32/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0859 - accuracy: 0.9774 - val_l
oss: 0.1156 - val_accuracy: 0.9668
Epoch 33/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.0836 - accuracy: 0.9779 - val_l
oss: 0.1139 - val_accuracy: 0.9678
Epoch 34/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.0811 - accuracy: 0.9785 - val_l
oss: 0.1130 - val_accuracy: 0.9676
Epoch 35/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.0791 - accuracy: 0.9793 - val_l
oss: 0.1107 - val_accuracy: 0.9676
Epoch 36/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.0771 - accuracy: 0.9799 - val_l
oss: 0.1096 - val_accuracy: 0.9698
Epoch 37/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.0750 - accuracy: 0.9803 - val_l
oss: 0.1087 - val_accuracy: 0.9690
Epoch 38/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.0730 - accuracy: 0.9814 - val_l
oss: 0.1070 - val_accuracy: 0.9712
Epoch 39/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.0712 - accuracy: 0.9816 - val_l
oss: 0.1059 - val_accuracy: 0.9692
Epoch 40/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.0694 - accuracy: 0.9823 - val_l
oss: 0.1052 - val_accuracy: 0.9704
Epoch 41/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.0675 - accuracy: 0.9828 - val_l
```

```
oss: 0.1040 - val_accuracy: 0.9692
Epoch 42/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.0662 - accuracy: 0.9830 - val_l
oss: 0.1029 - val_accuracy: 0.9696
Epoch 43/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.0645 - accuracy: 0.9837 - val_l
oss: 0.1034 - val_accuracy: 0.9706
Epoch 44/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.0631 - accuracy: 0.9839 - val_l
oss: 0.1012 - val_accuracy: 0.9696
Epoch 45/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0615 - accuracy: 0.9846 - val_l
oss: 0.1010 - val_accuracy: 0.9712
Epoch 46/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.0600 - accuracy: 0.9851 - val_l
oss: 0.1006 - val_accuracy: 0.9714
Epoch 47/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0589 - accuracy: 0.9854 - val_l
oss: 0.0986 - val_accuracy: 0.9708
Epoch 48/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.0575 - accuracy: 0.9858 - val_l
oss: 0.0981 - val_accuracy: 0.9704
Epoch 49/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.0562 - accuracy: 0.9855 - val_l
oss: 0.0965 - val_accuracy: 0.9710
Epoch 50/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.0549 - accuracy: 0.9868 - val_l
oss: 0.0960 - val_accuracy: 0.9726
Epoch 51/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.0538 - accuracy: 0.9868 - val_l
oss: 0.0957 - val_accuracy: 0.9706
Epoch 52/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.0526 - accuracy: 0.9872 - val_l
oss: 0.0954 - val_accuracy: 0.9714
Epoch 53/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.0514 - accuracy: 0.9878 - val_l
oss: 0.0948 - val_accuracy: 0.9722
Epoch 54/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.0503 - accuracy: 0.9879 - val_l
oss: 0.0953 - val_accuracy: 0.9728
Epoch 55/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0493 - accuracy: 0.9883 - val_l
oss: 0.0934 - val_accuracy: 0.9722
Epoch 56/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0481 - accuracy: 0.9888 - val_l
oss: 0.0921 - val_accuracy: 0.9738
Epoch 57/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.0472 - accuracy: 0.9890 - val_l
oss: 0.0923 - val_accuracy: 0.9738
Epoch 58/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.0463 - accuracy: 0.9894 - val_l
oss: 0.0925 - val_accuracy: 0.9738
Epoch 59/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0454 - accuracy: 0.9896 - val_l
oss: 0.0916 - val_accuracy: 0.9734
Epoch 60/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.0444 - accuracy: 0.9900 - val_l
oss: 0.0910 - val_accuracy: 0.9732
0:01:41.659543
```

In [62]:

```
pd.DataFrame(history.history).plot(figsize=(8, 5))
plt.grid(True)
plt.gca().set_ylim(0, 1)
plt.show()
```



In [63]:

```
y_pred=model.predict_classes(X_test)
y_pred
```

```
WARNING:tensorflow:Model was constructed with shape (None, 28, 28) for input KerasTensor(type_spec
=TensorSpec(shape=(None, 28, 28), dtype=tf.float32, name='flatten_input'), name='flatten_input', d
escription="created by layer 'flatten_input'"), but it was called on an input with incompatible sh
ape (32, 784).
```

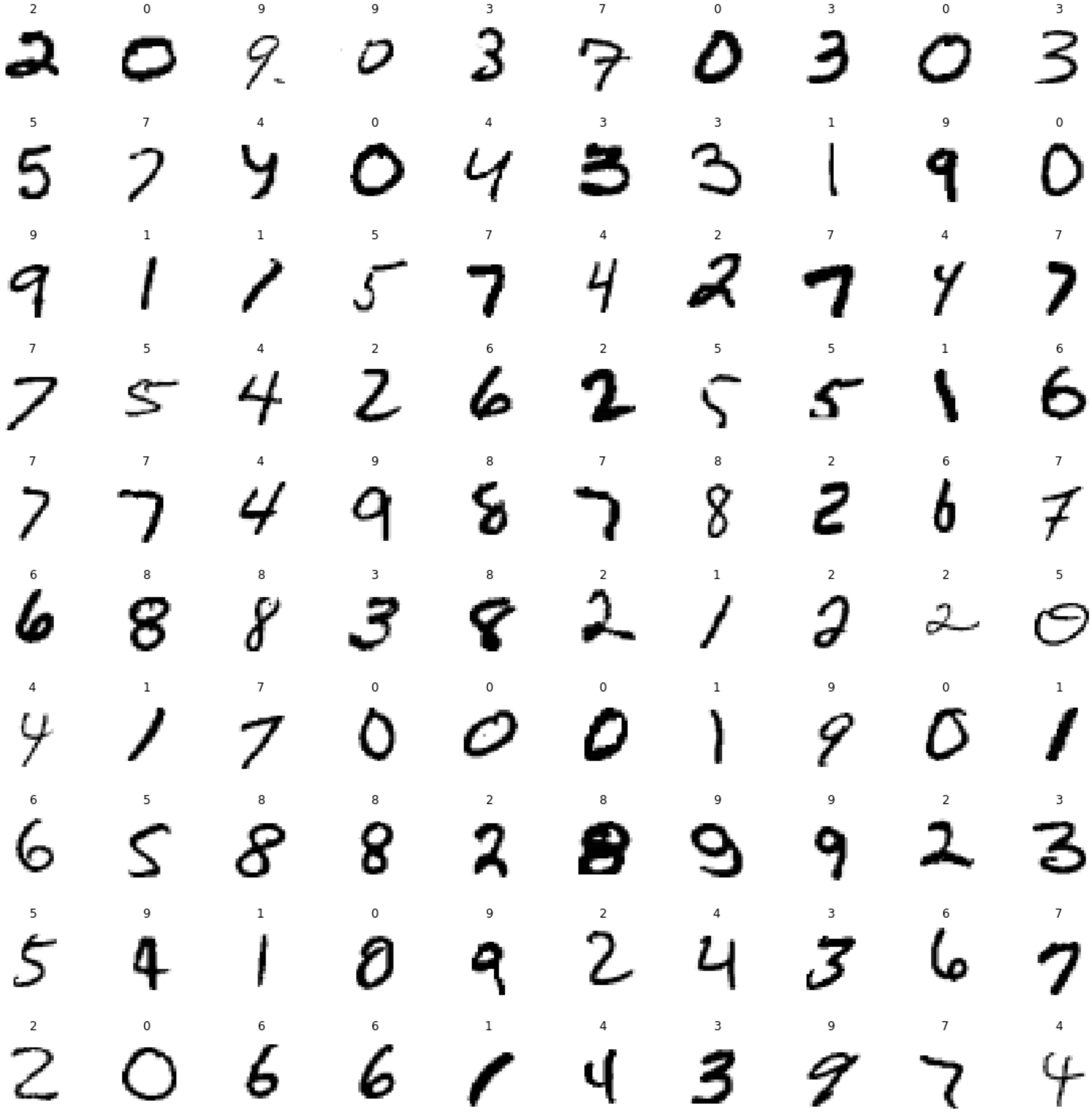
```
C:\Users\rocchm1\AppData\Roaming\Python\Python37\site-packages\tensorflow\python\keras\engine\sequ
ential.py:455: UserWarning: `model.predict_classes()` is deprecated and will be removed after 2021
-01-01. Please use instead:* `np.argmax(model.predict(x), axis=-1)`,   if your model does multi-cl
ass classification   (e.g. if it uses a `softmax` last-layer activation).* `(model.predict(x) > 0.
5).astype("int32")`,   if your model does binary classification   (e.g. if it uses a `sigmoid` las
t-layer activation).
  warnings.warn('`model.predict_classes()` is deprecated and '
```

Out[63]:

```
array([2, 0, 9, ..., 3, 9, 2], dtype=int64)
```

In [64]:

```
plt.figure(figsize=(20,20))
for index, image in enumerate(X_test[:100].reshape(100, 28, 28)):
    plt.subplot(10, 10, index + 1)
    plt.imshow(image, cmap="binary", interpolation="nearest")
    plt.axis('off')
    plt.title(y_pred[index], fontsize=12)
plt.subplots_adjust(wspace=0.2, hspace=0.5)
plt.show()
```



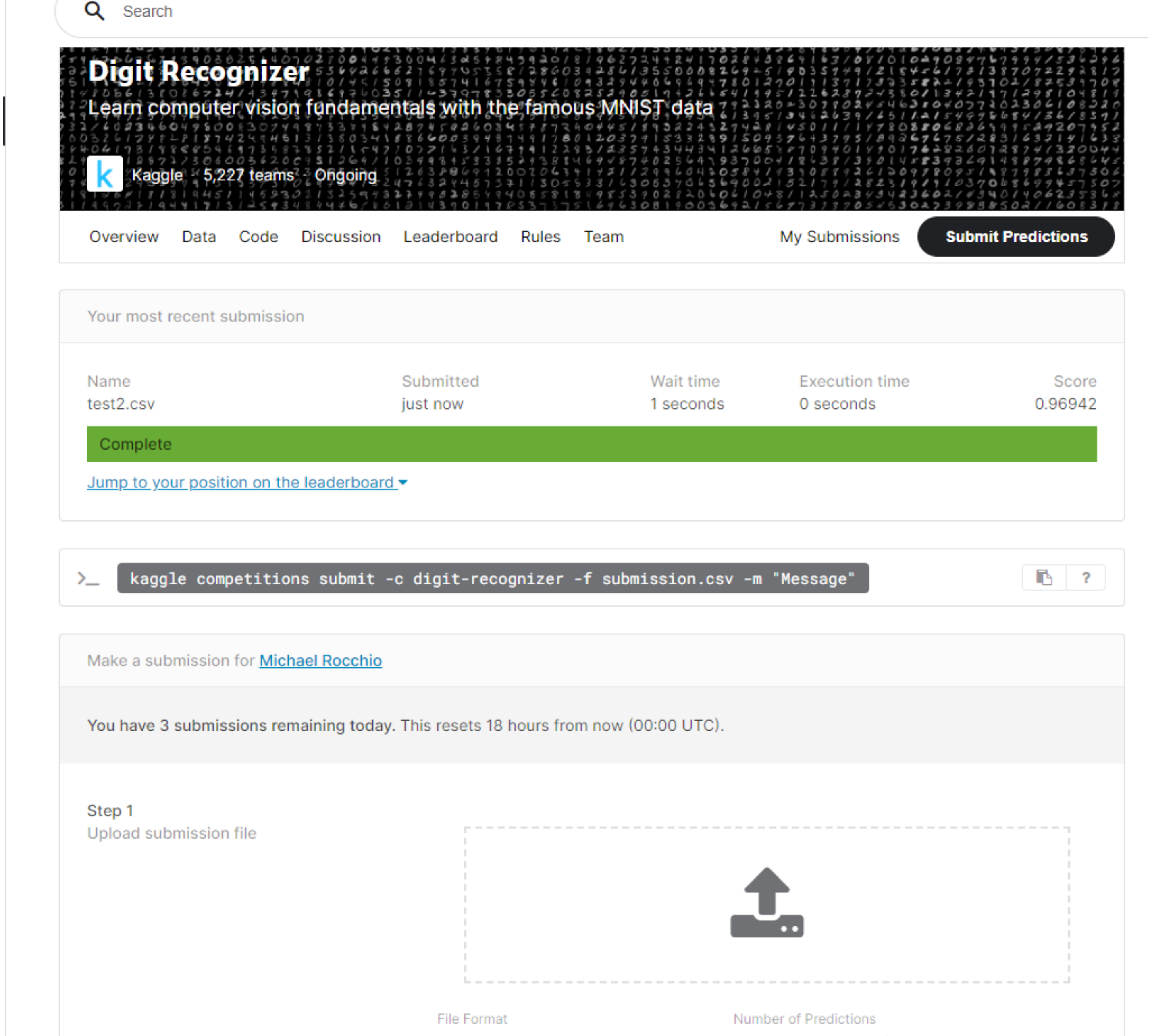
In [65]:

```
sample_sub['Label']=y_pred
sample_sub.to_csv('test2.csv')
```

In [93]:

```
from IPython.display import Image
Image(filename='test2.png')
######
######  MY USERNAME IN KAGGLE IS michaelrocchio ######
######
```

Out[93]:



Digit Recognizer

Learn computer vision fundamentals with the famous MNIST data

Kaggle · 5,227 teams · Ongoing

Your most recent submission

Name	Submitted	Wait time	Execution time	Score
test2.csv	just now	1 seconds	0 seconds	0.96942

Complete

Jump to your position on the leaderboard

```
kaggle competitions submit -c digit-recognizer -f submission.csv -m "Message"
```

Make a submission for Michael Rocchio

You have 3 submissions remaining today. This resets 18 hours from now (00:00 UTC).

Step 1

Upload submission file

File Format

Number of Predictions

In [77]:

```python
model = keras.models.Sequential()
model.add(keras.layers.Flatten(input_shape=[28, 28]))
model.add(keras.layers.Dense(100, activation="relu"))
model.add(keras.layers.Dense(100, activation="relu"))
model.add(keras.layers.Dense(100, activation="relu"))
model.add(keras.layers.Dense(100, activation="relu"))
model.add(keras.layers.Dense(10, activation="softmax"))
keras.backend.clear_session()
np.random.seed(42)
tf.random.set_seed(42)
model = keras.models.Sequential([
    keras.layers.Flatten(input_shape=[28, 28]),
    keras.layers.Dense(100, activation="relu"),
    keras.layers.Dense(100, activation="relu"),
    keras.layers.Dense(100, activation="relu"),
    keras.layers.Dense(100, activation="relu"),
    keras.layers.Dense(10, activation="softmax")
])
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
```

```
Layer (type)                 Output Shape              Param #
=================================================================
flatten (Flatten)            (None, 784)               0
_________________________________________________________________
dense (Dense)                (None, 100)               78500
_________________________________________________________________
dense_1 (Dense)              (None, 100)               10100
_________________________________________________________________
dense_2 (Dense)              (None, 100)               10100
_________________________________________________________________
dense_3 (Dense)              (None, 100)               10100
_________________________________________________________________
dense_4 (Dense)              (None, 10)                1010
=================================================================
Total params: 109,810
Trainable params: 109,810
Non-trainable params: 0
_________________________________________________________________
```

In [78]:

```
model.compile(loss="sparse_categorical_crossentropy",
              optimizer="sgd",
              metrics=["accuracy"])
```

In [79]:

```
start=datetime.now()
history = model.fit(X_train, y_train, epochs=60, validation_data=(X_valid, y_valid))
end=datetime.now()
print(end-start)
```

```
Epoch 1/60
1157/1157 [==============================] - 2s 2ms/step - loss: 1.0220 - accuracy: 0.6947 - val_l
oss: 0.4774 - val_accuracy: 0.8404
Epoch 2/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.3302 - accuracy: 0.9032 - val_l
oss: 0.3065 - val_accuracy: 0.9126
Epoch 3/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.2614 - accuracy: 0.9220 - val_l
oss: 0.3199 - val_accuracy: 0.9080
Epoch 4/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.2181 - accuracy: 0.9342 - val_l
oss: 0.2369 - val_accuracy: 0.9324
Epoch 5/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.1857 - accuracy: 0.9444 - val_l
oss: 0.2066 - val_accuracy: 0.9374
Epoch 6/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.1601 - accuracy: 0.9529 - val_l
oss: 0.1659 - val_accuracy: 0.9512
Epoch 7/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.1415 - accuracy: 0.9586 - val_l
oss: 0.1524 - val_accuracy: 0.9550
Epoch 8/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.1255 - accuracy: 0.9631 - val_l
oss: 0.1470 - val_accuracy: 0.9590
Epoch 9/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.1138 - accuracy: 0.9661 - val_l
oss: 0.1453 - val_accuracy: 0.9546
Epoch 10/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.1028 - accuracy: 0.9701 - val_l
oss: 0.1226 - val_accuracy: 0.9646
Epoch 11/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0929 - accuracy: 0.9728 - val_l
oss: 0.1202 - val_accuracy: 0.9664
Epoch 12/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.0854 - accuracy: 0.9744 - val_l
oss: 0.1271 - val_accuracy: 0.9640
Epoch 13/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0785 - accuracy: 0.9767 - val_l
oss: 0.1255 - val_accuracy: 0.9652
```

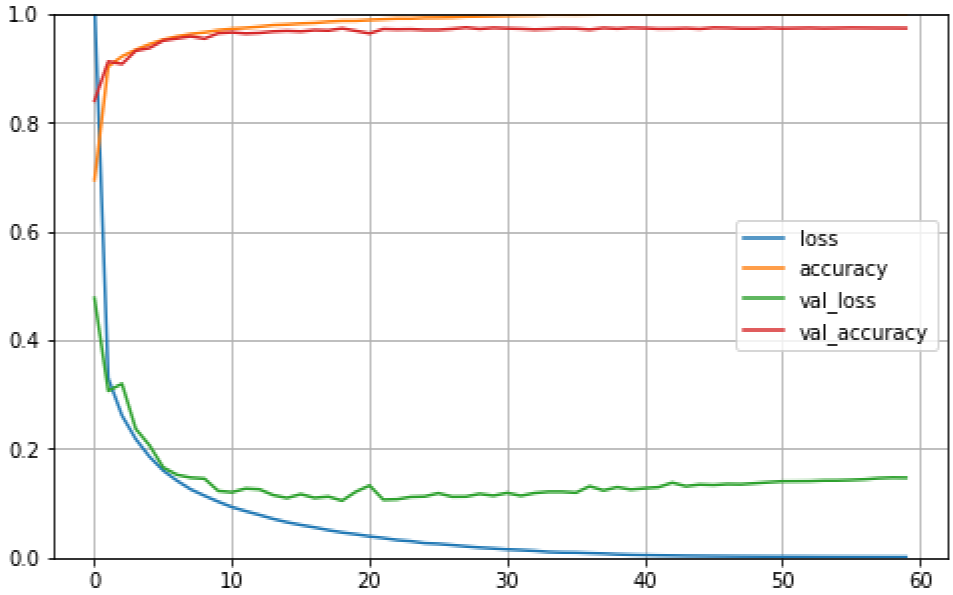
```
Epoch 14/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.0712 - accuracy: 0.9795 - val_l
oss: 0.1149 - val_accuracy: 0.9678
Epoch 15/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.0651 - accuracy: 0.9807 - val_l
oss: 0.1096 - val_accuracy: 0.9690
Epoch 16/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0600 - accuracy: 0.9824 - val_l
oss: 0.1167 - val_accuracy: 0.9678
Epoch 17/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.0555 - accuracy: 0.9835 - val_l
oss: 0.1098 - val_accuracy: 0.9704
Epoch 18/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.0506 - accuracy: 0.9858 - val_l
oss: 0.1123 - val_accuracy: 0.9696
Epoch 19/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0462 - accuracy: 0.9873 - val_l
oss: 0.1047 - val_accuracy: 0.9736
Epoch 20/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0429 - accuracy: 0.9874 - val_l
oss: 0.1212 - val_accuracy: 0.9690
Epoch 21/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.0390 - accuracy: 0.9890 - val_l
oss: 0.1326 - val_accuracy: 0.9642
Epoch 22/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.0359 - accuracy: 0.9902 - val_l
oss: 0.1064 - val_accuracy: 0.9724
Epoch 23/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.0323 - accuracy: 0.9916 - val_l
oss: 0.1072 - val_accuracy: 0.9716
Epoch 24/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.0301 - accuracy: 0.9916 - val_l
oss: 0.1117 - val_accuracy: 0.9720
Epoch 25/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.0266 - accuracy: 0.9930 - val_l
oss: 0.1125 - val_accuracy: 0.9708
Epoch 26/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.0253 - accuracy: 0.9932 - val_l
oss: 0.1185 - val_accuracy: 0.9708
Epoch 27/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0230 - accuracy: 0.9938 - val_l
oss: 0.1120 - val_accuracy: 0.9726
Epoch 28/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.0206 - accuracy: 0.9951 - val_l
oss: 0.1122 - val_accuracy: 0.9752
Epoch 29/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.0184 - accuracy: 0.9956 - val_l
oss: 0.1173 - val_accuracy: 0.9726
Epoch 30/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.0169 - accuracy: 0.9962 - val_l
oss: 0.1138 - val_accuracy: 0.9746
Epoch 31/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.0150 - accuracy: 0.9968 - val_l
oss: 0.1191 - val_accuracy: 0.9736
Epoch 32/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0139 - accuracy: 0.9974 - val_l
oss: 0.1136 - val_accuracy: 0.9728
Epoch 33/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.0124 - accuracy: 0.9978 - val_l
oss: 0.1188 - val_accuracy: 0.9714
Epoch 34/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.0103 - accuracy: 0.9986 - val_l
oss: 0.1211 - val_accuracy: 0.9724
Epoch 35/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0094 - accuracy: 0.9986 - val_l
oss: 0.1211 - val_accuracy: 0.9738
Epoch 36/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.0092 - accuracy: 0.9985 - val_l
oss: 0.1194 - val_accuracy: 0.9734
Epoch 37/60
```

```
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0082 - accuracy: 0.9988 - val_l
oss: 0.1313 - val_accuracy: 0.9712
Epoch 38/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.0073 - accuracy: 0.9991 - val_l
oss: 0.1236 - val_accuracy: 0.9744
Epoch 39/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.0060 - accuracy: 0.9995 - val_l
oss: 0.1296 - val_accuracy: 0.9730
Epoch 40/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.0052 - accuracy: 0.9995 - val_l
oss: 0.1252 - val_accuracy: 0.9746
Epoch 41/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.0046 - accuracy: 0.9998 - val_l
oss: 0.1276 - val_accuracy: 0.9740
Epoch 42/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.0041 - accuracy: 0.9998 - val_l
oss: 0.1294 - val_accuracy: 0.9728
Epoch 43/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0037 - accuracy: 0.9997 - val_l
oss: 0.1379 - val_accuracy: 0.9730
Epoch 44/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.0032 - accuracy: 0.9999 - val_l
oss: 0.1312 - val_accuracy: 0.9738
Epoch 45/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.0030 - accuracy: 0.9999 - val_l
oss: 0.1345 - val_accuracy: 0.9726
Epoch 46/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0027 - accuracy: 0.9999 - val_l
oss: 0.1335 - val_accuracy: 0.9748
Epoch 47/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.0025 - accuracy: 1.0000 - val_l
oss: 0.1353 - val_accuracy: 0.9744
Epoch 48/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.0023 - accuracy: 1.0000 - val_l
oss: 0.1348 - val_accuracy: 0.9734
Epoch 49/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.0022 - accuracy: 1.0000 - val_l
oss: 0.1367 - val_accuracy: 0.9734
Epoch 50/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.0020 - accuracy: 1.0000 - val_l
oss: 0.1386 - val_accuracy: 0.9744
Epoch 51/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.0019 - accuracy: 1.0000 - val_l
oss: 0.1403 - val_accuracy: 0.9736
Epoch 52/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.0018 - accuracy: 1.0000 - val_l
oss: 0.1404 - val_accuracy: 0.9740
Epoch 53/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.0017 - accuracy: 1.0000 - val_l
oss: 0.1406 - val_accuracy: 0.9744
Epoch 54/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0015 - accuracy: 1.0000 - val_l
oss: 0.1418 - val_accuracy: 0.9738
Epoch 55/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0015 - accuracy: 1.0000 - val_l
oss: 0.1419 - val_accuracy: 0.9742
Epoch 56/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.0014 - accuracy: 1.0000 - val_l
oss: 0.1427 - val_accuracy: 0.9746
Epoch 57/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.0013 - accuracy: 1.0000 - val_l
oss: 0.1440 - val_accuracy: 0.9744
Epoch 58/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.0013 - accuracy: 1.0000 - val_l
oss: 0.1461 - val_accuracy: 0.9742
Epoch 59/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.0012 - accuracy: 1.0000 - val_l
oss: 0.1471 - val_accuracy: 0.9742
Epoch 60/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.0012 - accuracy: 1.0000 - val_l
```

```
oss: 0.1468 - val_accuracy: 0.9740
0:01:42.446915
```

In [80]:

```python
pd.DataFrame(history.history).plot(figsize=(8, 5))
plt.grid(True)
plt.gca().set_ylim(0, 1)
plt.show()
```



In [81]:

```python
y_pred=model.predict_classes(X_test)
y_pred
```

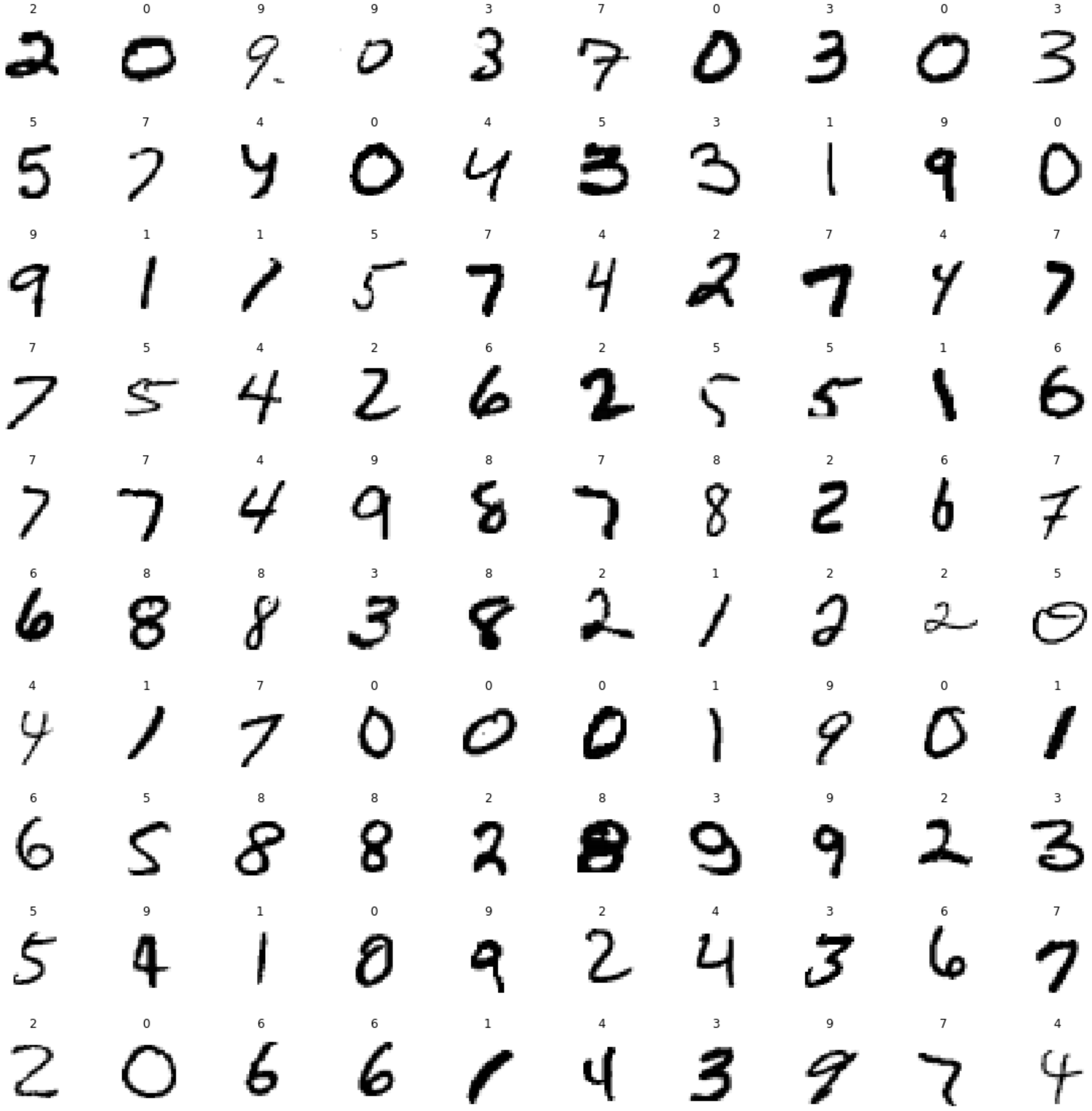
```
WARNING:tensorflow:Model was constructed with shape (None, 28, 28) for input KerasTensor(type_spec
=TensorSpec(shape=(None, 28, 28), dtype=tf.float32, name='flatten_input'), name='flatten_input', d
escription="created by layer 'flatten_input'"), but it was called on an input with incompatible sh
ape (32, 784).
C:\Users\rocchm1\AppData\Roaming\Python\Python37\site-packages\tensorflow\python\keras\engine\sequ
ential.py:455: UserWarning: `model.predict_classes()` is deprecated and will be removed after 2021
-01-01. Please use instead:* `np.argmax(model.predict(x), axis=-1)`,   if your model does multi-cl
ass classification   (e.g. if it uses a `softmax` last-layer activation).* `(model.predict(x) > 0.
5).astype("int32")`,   if your model does binary classification   (e.g. if it uses a `sigmoid` las
t-layer activation).
  warnings.warn('`model.predict_classes()` is deprecated and '
```

Out[81]:

```
array([2, 0, 9, ..., 3, 9, 2], dtype=int64)
```

In [82]:

```python
plt.figure(figsize=(20,20))
for index, image in enumerate(X_test[:100].reshape(100, 28, 28)):
    plt.subplot(10, 10, index + 1)
    plt.imshow(image, cmap="binary", interpolation="nearest")
    plt.axis('off')
    plt.title(y_pred[index], fontsize=12)
plt.subplots_adjust(wspace=0.2, hspace=0.5)
plt.show()
```



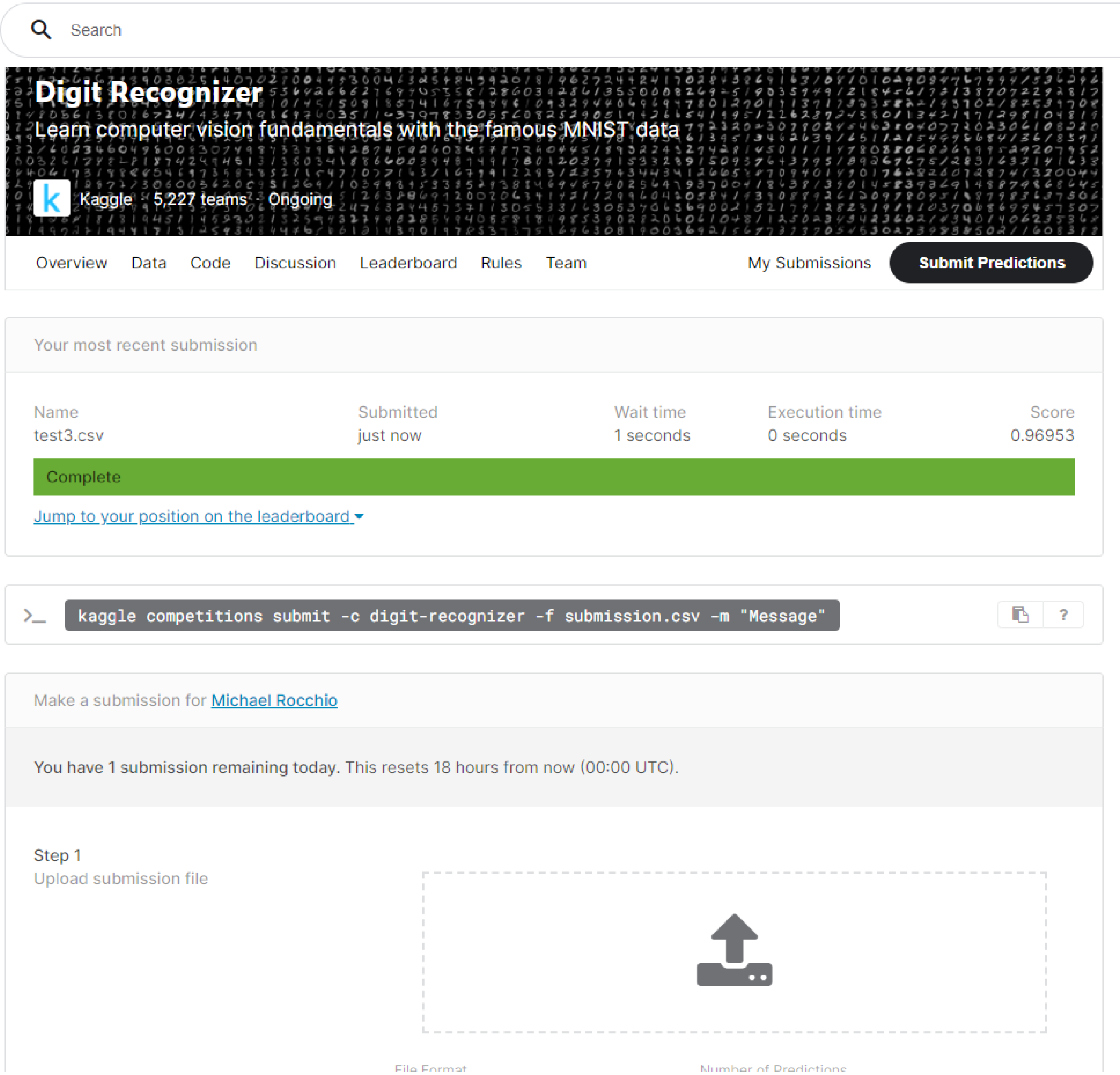
In [83]:

```
sample_sub['Label']=y_pred
sample_sub.to_csv('test3.csv')
```

In [103…

```
from IPython.display import Image
Image(filename='test3.png')
######
######  MY USERNAME IN KAGGLE IS michaelrocchio ######
######
```

Out[103…



Digit Recognizer

Learn computer vision fundamentals with the famous MNIST data

Kaggle · 5,227 teams · Ongoing

Your most recent submission

Name	Submitted	Wait time	Execution time	Score
test3.csv	just now	1 seconds	0 seconds	0.96953

Complete

Jump to your position on the leaderboard

```
kaggle competitions submit -c digit-recognizer -f submission.csv -m "Message"
```

Make a submission for Michael Rocchio

You have 1 submission remaining today. This resets 18 hours from now (00:00 UTC).

Step 1

Upload submission file

File Format

Number of Predictions

In [84]:

```
model = keras.models.Sequential()
model.add(keras.layers.Flatten(input_shape=[28, 28]))
model.add(keras.layers.Dense(100, activation="relu"))
model.add(keras.layers.Dense(100, activation="relu"))
model.add(keras.layers.Dense(100, activation="relu"))
model.add(keras.layers.Dense(100, activation="relu"))
model.add(keras.layers.Dense(10, activation="softmax"))
model.add(keras.layers.Dense(10, activation="softmax"))
keras.backend.clear_session()
np.random.seed(42)
tf.random.set_seed(42)
model = keras.models.Sequential([
    keras.layers.Flatten(input_shape=[28, 28]),
    keras.layers.Dense(100, activation="relu"),
    keras.layers.Dense(100, activation="relu"),
    keras.layers.Dense(100, activation="relu"),
    keras.layers.Dense(100, activation="relu"),
    keras.layers.Dense(10, activation="softmax"),
    keras.layers.Dense(10, activation="softmax")
])
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten (Flatten)            (None, 784)               0
_________________________________________________________________
dense (Dense)                (None, 100)               78500
_________________________________________________________________
dense_1 (Dense)              (None, 100)               10100
_________________________________________________________________
dense_2 (Dense)              (None, 100)               10100
_________________________________________________________________
dense_3 (Dense)              (None, 100)               10100
_________________________________________________________________
dense_4 (Dense)              (None, 10)                1010
=================================================================
Total params: 109,810
Trainable params: 109,810
Non-trainable params: 0
_________________________________________________________________
```

In [85]:

```
model.compile(loss="sparse_categorical_crossentropy",
              optimizer="sgd",
              metrics=["accuracy"])
```

In [86]:

```
start=datetime.now()
history = model.fit(X_train, y_train, epochs=60, validation_data=(X_valid, y_valid))
end=datetime.now()
print(end-start)
```

```
Epoch 1/60
1157/1157 [==============================] - 2s 2ms/step - loss: 1.0220 - accuracy: 0.6947 - val_l
oss: 0.4774 - val_accuracy: 0.8404
Epoch 2/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.3302 - accuracy: 0.9032 - val_l
oss: 0.3065 - val_accuracy: 0.9126
Epoch 3/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.2614 - accuracy: 0.9220 - val_l
oss: 0.3199 - val_accuracy: 0.9080
Epoch 4/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.2181 - accuracy: 0.9342 - val_l
oss: 0.2369 - val_accuracy: 0.9324
Epoch 5/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.1857 - accuracy: 0.9444 - val_l
oss: 0.2066 - val_accuracy: 0.9374
Epoch 6/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.1601 - accuracy: 0.9529 - val_l
oss: 0.1659 - val_accuracy: 0.9512
Epoch 7/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.1415 - accuracy: 0.9586 - val_l
oss: 0.1524 - val_accuracy: 0.9550
Epoch 8/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.1255 - accuracy: 0.9631 - val_l
oss: 0.1470 - val_accuracy: 0.9590
Epoch 9/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.1138 - accuracy: 0.9661 - val_l
oss: 0.1453 - val_accuracy: 0.9546
Epoch 10/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.1028 - accuracy: 0.9701 - val_l
oss: 0.1226 - val_accuracy: 0.9646
Epoch 11/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.0929 - accuracy: 0.9728 - val_l
oss: 0.1202 - val_accuracy: 0.9664
Epoch 12/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.0854 - accuracy: 0.9744 - val_l
oss: 0.1271 - val_accuracy: 0.9640
Epoch 13/60
```

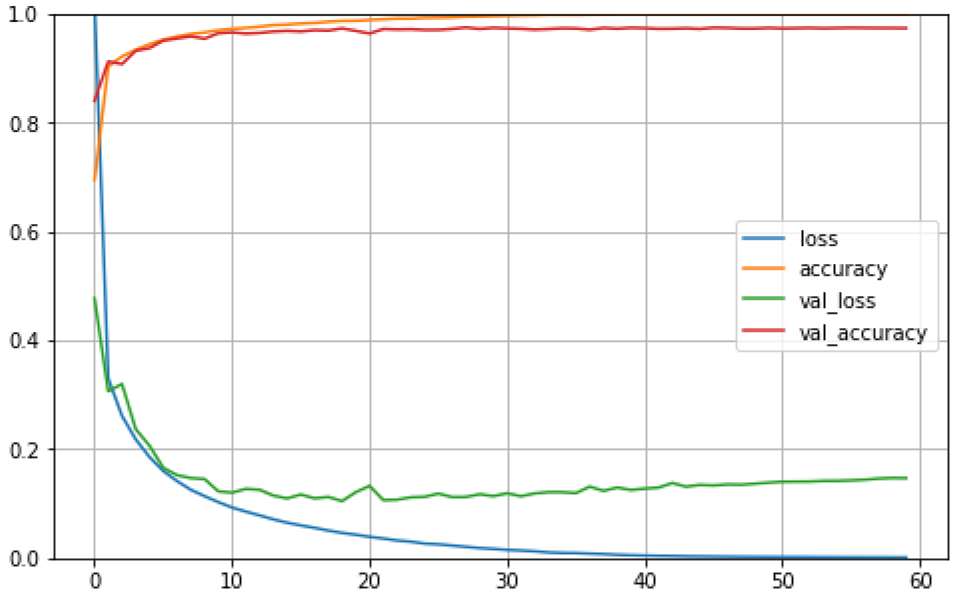
```
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0785 - accuracy: 0.9767 - val_l
oss: 0.1255 - val_accuracy: 0.9652
Epoch 14/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0712 - accuracy: 0.9795 - val_l
oss: 0.1149 - val_accuracy: 0.9678
Epoch 15/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0651 - accuracy: 0.9807 - val_l
oss: 0.1096 - val_accuracy: 0.9690
Epoch 16/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0600 - accuracy: 0.9824 - val_l
oss: 0.1167 - val_accuracy: 0.9678
Epoch 17/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0555 - accuracy: 0.9835 - val_l
oss: 0.1098 - val_accuracy: 0.9704
Epoch 18/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0506 - accuracy: 0.9858 - val_l
oss: 0.1123 - val_accuracy: 0.9696
Epoch 19/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0462 - accuracy: 0.9873 - val_l
oss: 0.1047 - val_accuracy: 0.9736
Epoch 20/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0429 - accuracy: 0.9874 - val_l
oss: 0.1212 - val_accuracy: 0.9690
Epoch 21/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0390 - accuracy: 0.9890 - val_l
oss: 0.1326 - val_accuracy: 0.9642
Epoch 22/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0359 - accuracy: 0.9902 - val_l
oss: 0.1064 - val_accuracy: 0.9724
Epoch 23/60
1157/1157 [==============================] - 3s 2ms/step - loss: 0.0323 - accuracy: 0.9916 - val_l
oss: 0.1072 - val_accuracy: 0.9716
Epoch 24/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0301 - accuracy: 0.9916 - val_l
oss: 0.1117 - val_accuracy: 0.9720
Epoch 25/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0266 - accuracy: 0.9930 - val_l
oss: 0.1125 - val_accuracy: 0.9708
Epoch 26/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.0253 - accuracy: 0.9932 - val_l
oss: 0.1185 - val_accuracy: 0.9708
Epoch 27/60
1157/1157 [==============================] - 2s 1ms/step - loss: 0.0230 - accuracy: 0.9938 - val_l
oss: 0.1120 - val_accuracy: 0.9726
Epoch 28/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0206 - accuracy: 0.9951 - val_l
oss: 0.1122 - val_accuracy: 0.9752
Epoch 29/60
1157/1157 [==============================] - 4s 3ms/step - loss: 0.0184 - accuracy: 0.9956 - val_l
oss: 0.1173 - val_accuracy: 0.9726
Epoch 30/60
1157/1157 [==============================] - 3s 3ms/step - loss: 0.0169 - accuracy: 0.9962 - val_l
oss: 0.1138 - val_accuracy: 0.9746
Epoch 31/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0150 - accuracy: 0.9968 - val_l
oss: 0.1191 - val_accuracy: 0.9736
Epoch 32/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0139 - accuracy: 0.9974 - val_l
oss: 0.1136 - val_accuracy: 0.9728
Epoch 33/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0124 - accuracy: 0.9978 - val_l
oss: 0.1188 - val_accuracy: 0.9714
Epoch 34/60
1157/1157 [==============================] - 4s 3ms/step - loss: 0.0103 - accuracy: 0.9986 - val_l
oss: 0.1211 - val_accuracy: 0.9724
Epoch 35/60
1157/1157 [==============================] - 3s 3ms/step - loss: 0.0094 - accuracy: 0.9986 - val_l
oss: 0.1211 - val_accuracy: 0.9738
Epoch 36/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0092 - accuracy: 0.9985 - val_l
```

```
oss: 0.1194 - val_accuracy: 0.9734
Epoch 37/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0082 - accuracy: 0.9988 - val_l
oss: 0.1313 - val_accuracy: 0.9712
Epoch 38/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0073 - accuracy: 0.9991 - val_l
oss: 0.1236 - val_accuracy: 0.9744
Epoch 39/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0060 - accuracy: 0.9995 - val_l
oss: 0.1296 - val_accuracy: 0.9730
Epoch 40/60
1157/1157 [==============================] - 3s 2ms/step - loss: 0.0052 - accuracy: 0.9995 - val_l
oss: 0.1252 - val_accuracy: 0.9746
Epoch 41/60
1157/1157 [==============================] - 3s 2ms/step - loss: 0.0046 - accuracy: 0.9998 - val_l
oss: 0.1276 - val_accuracy: 0.9740
Epoch 42/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0041 - accuracy: 0.9998 - val_l
oss: 0.1294 - val_accuracy: 0.9728
Epoch 43/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0037 - accuracy: 0.9997 - val_l
oss: 0.1379 - val_accuracy: 0.9730
Epoch 44/60
1157/1157 [==============================] - 3s 3ms/step - loss: 0.0032 - accuracy: 0.9999 - val_l
oss: 0.1312 - val_accuracy: 0.9738
Epoch 45/60
1157/1157 [==============================] - 3s 3ms/step - loss: 0.0030 - accuracy: 0.9999 - val_l
oss: 0.1345 - val_accuracy: 0.9726
Epoch 46/60
1157/1157 [==============================] - 3s 2ms/step - loss: 0.0027 - accuracy: 0.9999 - val_l
oss: 0.1335 - val_accuracy: 0.9748
Epoch 47/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0025 - accuracy: 1.0000 - val_l
oss: 0.1353 - val_accuracy: 0.9744
Epoch 48/60
1157/1157 [==============================] - 3s 3ms/step - loss: 0.0023 - accuracy: 1.0000 - val_l
oss: 0.1348 - val_accuracy: 0.9734
Epoch 49/60
1157/1157 [==============================] - 3s 2ms/step - loss: 0.0022 - accuracy: 1.0000 - val_l
oss: 0.1367 - val_accuracy: 0.9734
Epoch 50/60
1157/1157 [==============================] - 3s 2ms/step - loss: 0.0020 - accuracy: 1.0000 - val_l
oss: 0.1386 - val_accuracy: 0.9744
Epoch 51/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0019 - accuracy: 1.0000 - val_l
oss: 0.1403 - val_accuracy: 0.9736
Epoch 52/60
1157/1157 [==============================] - 4s 4ms/step - loss: 0.0018 - accuracy: 1.0000 - val_l
oss: 0.1404 - val_accuracy: 0.9740
Epoch 53/60
1157/1157 [==============================] - 3s 3ms/step - loss: 0.0017 - accuracy: 1.0000 - val_l
oss: 0.1406 - val_accuracy: 0.9744
Epoch 54/60
1157/1157 [==============================] - 3s 3ms/step - loss: 0.0015 - accuracy: 1.0000 - val_l
oss: 0.1418 - val_accuracy: 0.9738
Epoch 55/60
1157/1157 [==============================] - 4s 3ms/step - loss: 0.0015 - accuracy: 1.0000 - val_l
oss: 0.1419 - val_accuracy: 0.9742
Epoch 56/60
1157/1157 [==============================] - 3s 3ms/step - loss: 0.0014 - accuracy: 1.0000 - val_l
oss: 0.1427 - val_accuracy: 0.9746
Epoch 57/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0013 - accuracy: 1.0000 - val_l
oss: 0.1440 - val_accuracy: 0.9744
Epoch 58/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0013 - accuracy: 1.0000 - val_l
oss: 0.1461 - val_accuracy: 0.9742
Epoch 59/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0012 - accuracy: 1.0000 - val_l
oss: 0.1471 - val_accuracy: 0.9742
```

```
Epoch 60/60
1157/1157 [==============================] - 2s 2ms/step - loss: 0.0012 - accuracy: 1.0000 - val_l
oss: 0.1468 - val_accuracy: 0.9740
0:02:20.549488
```

In [87]:

```
pd.DataFrame(history.history).plot(figsize=(8, 5))
plt.grid(True)
plt.gca().set_ylim(0, 1)
plt.show()
```



In [88]:

```
y_pred=model.predict_classes(X_test)
y_pred
```

```
WARNING:tensorflow:Model was constructed with shape (None, 28, 28) for input KerasTensor(type_spec
=TensorSpec(shape=(None, 28, 28), dtype=tf.float32, name='flatten_input'), name='flatten_input', d
escription="created by layer 'flatten_input'"), but it was called on an input with incompatible sh
ape (32, 784).
```

```
C:\Users\rocchm1\AppData\Roaming\Python\Python37\site-packages\tensorflow\python\keras\engine\sequ
ential.py:455: UserWarning: `model.predict_classes()` is deprecated and will be removed after 2021
-01-01. Please use instead:* `np.argmax(model.predict(x), axis=-1)`,   if your model does multi-cl
ass classification   (e.g. if it uses a `softmax` last-layer activation).* `(model.predict(x) > 0.
5).astype("int32")`,   if your model does binary classification   (e.g. if it uses a `sigmoid` las
t-layer activation).
  warnings.warn('`model.predict_classes()` is deprecated and '
```

Out[88]:

```
array([2, 0, 9, ..., 3, 9, 2], dtype=int64)
```

In [89]:

```
plt.figure(figsize=(20,20))
for index, image in enumerate(X_test[:100].reshape(100, 28, 28)):
    plt.subplot(10, 10, index + 1)
    plt.imshow(image, cmap="binary", interpolation="nearest")
    plt.axis('off')
    plt.title(y_pred[index], fontsize=12)
plt.subplots_adjust(wspace=0.2, hspace=0.5)
plt.show()
```

2 0 9 9 3 7 0 3 0 3
5 7 4 0 4 5 3 1 9 0
9 1 1 5 7 4 2 7 4 7
7 5 4 2 6 2 5 5 1 6
7 7 4 9 8 7 8 2 6 7
6 8 8 3 8 2 1 2 2 5
4 1 7 0 0 0 1 9 0 1
6 5 8 8 2 8 3 9 2 3
5 9 1 0 9 2 4 3 6 7
2 0 6 6 1 4 3 9 7 4

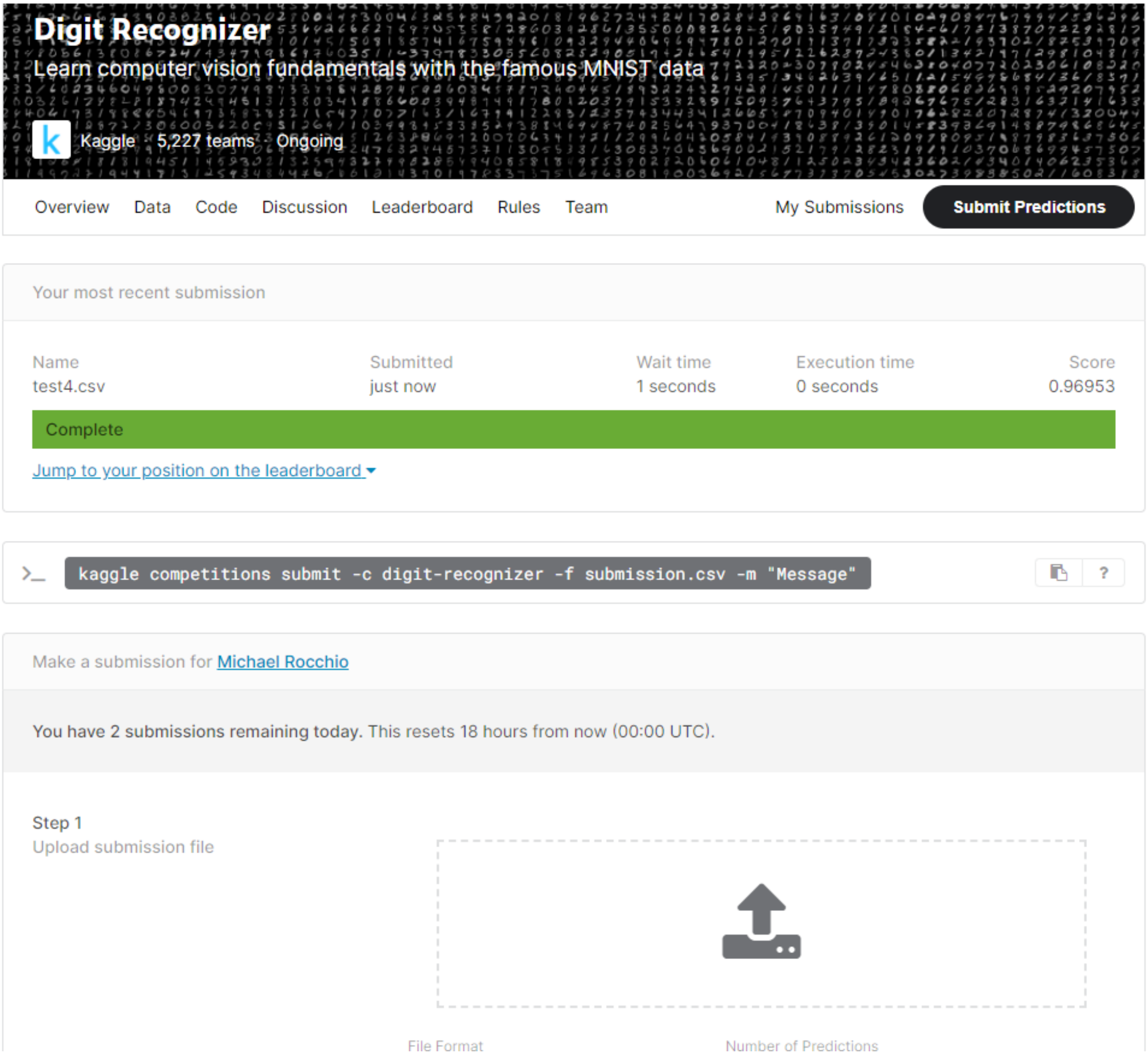
In [91]:

```
sample_sub['Label']=y_pred
sample_sub.to_csv('test4.csv', index=False)
```

In [95]:

```
from IPython.display import Image
Image(filename='test4.png')
######
######  MY USERNAME IN KAGGLE IS michaelrocchio ######
######
```

Out[95]:



Your most recent submission

Name	Submitted	Wait time	Execution time	Score
test4.csv	just now	1 seconds	0 seconds	0.96953

Complete

Jump to your position on the leaderboard ▾

```
kaggle competitions submit -c digit-recognizer -f submission.csv -m "Message"
```

Make a submission for Michael Rocchio

You have 2 submissions remaining today. This resets 18 hours from now (00:00 UTC).

Step 1
Upload submission file

In [96]:

```
model = keras.models.Sequential()
model.add(keras.layers.Flatten(input_shape=[28, 28]))
model.add(keras.layers.Dense(900, activation="relu"))
model.add(keras.layers.Dense(300, activation="relu"))
model.add(keras.layers.Dense(100, activation="relu"))
model.add(keras.layers.Dense(10, activation="softmax"))
keras.backend.clear_session()
np.random.seed(42)
tf.random.set_seed(42)
model = keras.models.Sequential([
    keras.layers.Flatten(input_shape=[28, 28]),
    keras.layers.Dense(900, activation="relu"),
    keras.layers.Dense(300, activation="relu"),
    keras.layers.Dense(100, activation="relu"),
    keras.layers.Dense(10, activation="softmax"),
])
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
flatten (Flatten)            (None, 784)               0         
_________________________________________________________________
```

```
dense (Dense)                (None, 900)               706500
_________________________________________________________________
dense_1 (Dense)              (None, 300)               270300
_________________________________________________________________
dense_2 (Dense)              (None, 100)               30100
_________________________________________________________________
dense_3 (Dense)              (None, 10)                1010
=================================================================
Total params: 1,007,910
Trainable params: 1,007,910
Non-trainable params: 0
_________________________________________________________________
```

In [97]:

```python
model.compile(loss="sparse_categorical_crossentropy",
              optimizer="sgd",
              metrics=["accuracy"])
```

In [98]:

```python
start=datetime.now()
history = model.fit(X_train, y_train, epochs=60, validation_data=(X_valid, y_valid))
end=datetime.now()
print(end-start)
```

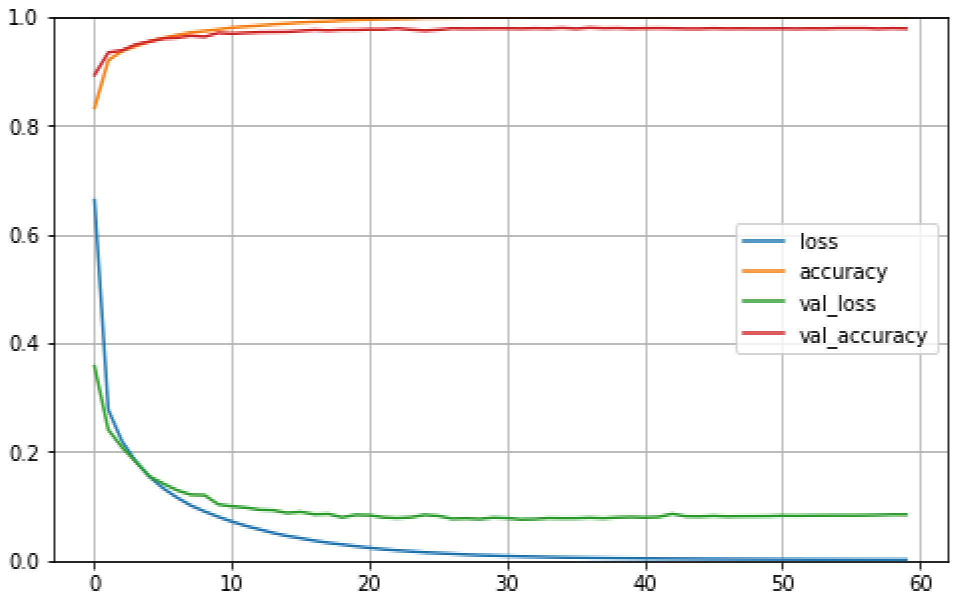
```
Epoch 1/60
1157/1157 [==============================] - 7s 6ms/step - loss: 0.6630 - accuracy: 0.8329 - val_l
oss: 0.3575 - val_accuracy: 0.8934
Epoch 2/60
1157/1157 [==============================] - 7s 6ms/step - loss: 0.2777 - accuracy: 0.9204 - val_l
oss: 0.2407 - val_accuracy: 0.9346
Epoch 3/60
1157/1157 [==============================] - 7s 6ms/step - loss: 0.2195 - accuracy: 0.9375 - val_l
oss: 0.2084 - val_accuracy: 0.9386
Epoch 4/60
1157/1157 [==============================] - 7s 6ms/step - loss: 0.1826 - accuracy: 0.9464 - val_l
oss: 0.1818 - val_accuracy: 0.9494
Epoch 5/60
1157/1157 [==============================] - 7s 6ms/step - loss: 0.1543 - accuracy: 0.9554 - val_l
oss: 0.1547 - val_accuracy: 0.9548
Epoch 6/60
1157/1157 [==============================] - 7s 6ms/step - loss: 0.1330 - accuracy: 0.9613 - val_l
oss: 0.1416 - val_accuracy: 0.9604
Epoch 7/60
1157/1157 [==============================] - 7s 6ms/step - loss: 0.1160 - accuracy: 0.9666 - val_l
oss: 0.1294 - val_accuracy: 0.9618
Epoch 8/60
1157/1157 [==============================] - 7s 6ms/step - loss: 0.1014 - accuracy: 0.9715 - val_l
oss: 0.1212 - val_accuracy: 0.9656
Epoch 9/60
1157/1157 [==============================] - 7s 6ms/step - loss: 0.0904 - accuracy: 0.9742 - val_l
oss: 0.1205 - val_accuracy: 0.9636
Epoch 10/60
1157/1157 [==============================] - 8s 7ms/step - loss: 0.0805 - accuracy: 0.9770 - val_l
oss: 0.1033 - val_accuracy: 0.9708
Epoch 11/60
1157/1157 [==============================] - 7s 6ms/step - loss: 0.0716 - accuracy: 0.9804 - val_l
oss: 0.0998 - val_accuracy: 0.9694
Epoch 12/60
1157/1157 [==============================] - 8s 7ms/step - loss: 0.0639 - accuracy: 0.9825 - val_l
oss: 0.0975 - val_accuracy: 0.9708
Epoch 13/60
1157/1157 [==============================] - 9s 8ms/step - loss: 0.0573 - accuracy: 0.9842 - val_l
oss: 0.0936 - val_accuracy: 0.9720
Epoch 14/60
1157/1157 [==============================] - 9s 8ms/step - loss: 0.0509 - accuracy: 0.9865 - val_l
oss: 0.0925 - val_accuracy: 0.9724
Epoch 15/60
1157/1157 [==============================] - 10s 8ms/step - loss: 0.0454 - accuracy: 0.9881 - val_
loss: 0.0876 - val_accuracy: 0.9728
```

```
Epoch 16/60
1157/1157 [==============================] - 9s 8ms/step - loss: 0.0411 - accuracy: 0.9899 - val_l
oss: 0.0896 - val_accuracy: 0.9744
Epoch 17/60
1157/1157 [==============================] - 9s 7ms/step - loss: 0.0365 - accuracy: 0.9914 - val_l
oss: 0.0850 - val_accuracy: 0.9762
Epoch 18/60
1157/1157 [==============================] - 9s 8ms/step - loss: 0.0326 - accuracy: 0.9920 - val_l
oss: 0.0858 - val_accuracy: 0.9750
Epoch 19/60
1157/1157 [==============================] - 8s 7ms/step - loss: 0.0296 - accuracy: 0.9935 - val_l
oss: 0.0796 - val_accuracy: 0.9764
Epoch 20/60
1157/1157 [==============================] - 8s 7ms/step - loss: 0.0263 - accuracy: 0.9946 - val_l
oss: 0.0841 - val_accuracy: 0.9762
Epoch 21/60
1157/1157 [==============================] - 8s 7ms/step - loss: 0.0234 - accuracy: 0.9953 - val_l
oss: 0.0834 - val_accuracy: 0.9772
Epoch 22/60
1157/1157 [==============================] - 10s 8ms/step - loss: 0.0209 - accuracy: 0.9961 - val_
loss: 0.0796 - val_accuracy: 0.9772
Epoch 23/60
1157/1157 [==============================] - 8s 7ms/step - loss: 0.0187 - accuracy: 0.9967 - val_l
oss: 0.0782 - val_accuracy: 0.9790
Epoch 24/60
1157/1157 [==============================] - 7s 6ms/step - loss: 0.0170 - accuracy: 0.9971 - val_l
oss: 0.0797 - val_accuracy: 0.9768
Epoch 25/60
1157/1157 [==============================] - 7s 6ms/step - loss: 0.0151 - accuracy: 0.9980 - val_l
oss: 0.0841 - val_accuracy: 0.9748
Epoch 26/60
1157/1157 [==============================] - 8s 7ms/step - loss: 0.0138 - accuracy: 0.9983 - val_l
oss: 0.0822 - val_accuracy: 0.9766
Epoch 27/60
1157/1157 [==============================] - 7s 6ms/step - loss: 0.0124 - accuracy: 0.9985 - val_l
oss: 0.0770 - val_accuracy: 0.9790
Epoch 28/60
1157/1157 [==============================] - 8s 7ms/step - loss: 0.0110 - accuracy: 0.9991 - val_l
oss: 0.0776 - val_accuracy: 0.9786
Epoch 29/60
1157/1157 [==============================] - 8s 7ms/step - loss: 0.0100 - accuracy: 0.9991 - val_l
oss: 0.0764 - val_accuracy: 0.9788
Epoch 30/60
1157/1157 [==============================] - 8s 7ms/step - loss: 0.0092 - accuracy: 0.9993 - val_l
oss: 0.0791 - val_accuracy: 0.9790
Epoch 31/60
1157/1157 [==============================] - 8s 7ms/step - loss: 0.0084 - accuracy: 0.9994 - val_l
oss: 0.0781 - val_accuracy: 0.9790
Epoch 32/60
1157/1157 [==============================] - 9s 8ms/step - loss: 0.0076 - accuracy: 0.9995 - val_l
oss: 0.0758 - val_accuracy: 0.9786
Epoch 33/60
1157/1157 [==============================] - 9s 8ms/step - loss: 0.0070 - accuracy: 0.9997 - val_l
oss: 0.0764 - val_accuracy: 0.9792
Epoch 34/60
1157/1157 [==============================] - 8s 7ms/step - loss: 0.0064 - accuracy: 0.9996 - val_l
oss: 0.0782 - val_accuracy: 0.9788
Epoch 35/60
1157/1157 [==============================] - 9s 7ms/step - loss: 0.0059 - accuracy: 0.9998 - val_l
oss: 0.0778 - val_accuracy: 0.9800
Epoch 36/60
1157/1157 [==============================] - 11s 9ms/step - loss: 0.0055 - accuracy: 0.9998 - val_
loss: 0.0779 - val_accuracy: 0.9784
Epoch 37/60
1157/1157 [==============================] - 7s 6ms/step - loss: 0.0051 - accuracy: 0.9998 - val_l
oss: 0.0789 - val_accuracy: 0.9804
Epoch 38/60
1157/1157 [==============================] - 9s 8ms/step - loss: 0.0048 - accuracy: 0.9999 - val_l
oss: 0.0778 - val_accuracy: 0.9792
Epoch 39/60
```

```
1157/1157 [==============================] - 8s 7ms/step - loss: 0.0044 - accuracy: 0.9999 - val_l
oss: 0.0794 - val_accuracy: 0.9796
Epoch 40/60
1157/1157 [==============================] - 10s 9ms/step - loss: 0.0042 - accuracy: 0.9999 - val_
loss: 0.0801 - val_accuracy: 0.9788
Epoch 41/60
1157/1157 [==============================] - 10s 9ms/step - loss: 0.0038 - accuracy: 1.0000 - val_
loss: 0.0794 - val_accuracy: 0.9792
Epoch 42/60
1157/1157 [==============================] - 8s 7ms/step - loss: 0.0036 - accuracy: 0.9999 - val_l
oss: 0.0802 - val_accuracy: 0.9794
Epoch 43/60
1157/1157 [==============================] - 9s 7ms/step - loss: 0.0034 - accuracy: 0.9999 - val_l
oss: 0.0860 - val_accuracy: 0.9790
Epoch 44/60
1157/1157 [==============================] - 8s 7ms/step - loss: 0.0032 - accuracy: 1.0000 - val_l
oss: 0.0814 - val_accuracy: 0.9784
Epoch 45/60
1157/1157 [==============================] - 7s 6ms/step - loss: 0.0030 - accuracy: 1.0000 - val_l
oss: 0.0811 - val_accuracy: 0.9784
Epoch 46/60
1157/1157 [==============================] - 8s 7ms/step - loss: 0.0029 - accuracy: 1.0000 - val_l
oss: 0.0823 - val_accuracy: 0.9792
Epoch 47/60
1157/1157 [==============================] - 8s 7ms/step - loss: 0.0027 - accuracy: 1.0000 - val_l
oss: 0.0812 - val_accuracy: 0.9786
Epoch 48/60
1157/1157 [==============================] - 7s 6ms/step - loss: 0.0026 - accuracy: 1.0000 - val_l
oss: 0.0815 - val_accuracy: 0.9788
Epoch 49/60
1157/1157 [==============================] - 7s 6ms/step - loss: 0.0025 - accuracy: 1.0000 - val_l
oss: 0.0817 - val_accuracy: 0.9786
Epoch 50/60
1157/1157 [==============================] - 8s 7ms/step - loss: 0.0023 - accuracy: 1.0000 - val_l
oss: 0.0819 - val_accuracy: 0.9786
Epoch 51/60
1157/1157 [==============================] - 7s 6ms/step - loss: 0.0023 - accuracy: 1.0000 - val_l
oss: 0.0827 - val_accuracy: 0.9788
Epoch 52/60
1157/1157 [==============================] - 7s 6ms/step - loss: 0.0022 - accuracy: 1.0000 - val_l
oss: 0.0825 - val_accuracy: 0.9784
Epoch 53/60
1157/1157 [==============================] - 7s 6ms/step - loss: 0.0021 - accuracy: 1.0000 - val_l
oss: 0.0829 - val_accuracy: 0.9788
Epoch 54/60
1157/1157 [==============================] - 7s 6ms/step - loss: 0.0020 - accuracy: 1.0000 - val_l
oss: 0.0830 - val_accuracy: 0.9786
Epoch 55/60
1157/1157 [==============================] - 7s 6ms/step - loss: 0.0019 - accuracy: 1.0000 - val_l
oss: 0.0832 - val_accuracy: 0.9794
Epoch 56/60
1157/1157 [==============================] - 7s 6ms/step - loss: 0.0018 - accuracy: 1.0000 - val_l
oss: 0.0832 - val_accuracy: 0.9794
Epoch 57/60
1157/1157 [==============================] - 7s 6ms/step - loss: 0.0018 - accuracy: 1.0000 - val_l
oss: 0.0834 - val_accuracy: 0.9794
Epoch 58/60
1157/1157 [==============================] - 8s 7ms/step - loss: 0.0017 - accuracy: 1.0000 - val_l
oss: 0.0839 - val_accuracy: 0.9784
Epoch 59/60
1157/1157 [==============================] - 10s 8ms/step - loss: 0.0017 - accuracy: 1.0000 - val_
loss: 0.0847 - val_accuracy: 0.9792
Epoch 60/60
1157/1157 [==============================] - 8s 7ms/step - loss: 0.0016 - accuracy: 1.0000 - val_l
oss: 0.0846 - val_accuracy: 0.9786
0:07:58.358673
```

In [99]:

```
pd.DataFrame(history.history).plot(figsize=(8, 5))
plt.grid(True)
plt.gca().set_ylim(0, 1)
plt.show()
```



In [100…

```
y_pred=model.predict_classes(X_test)
y_pred
```

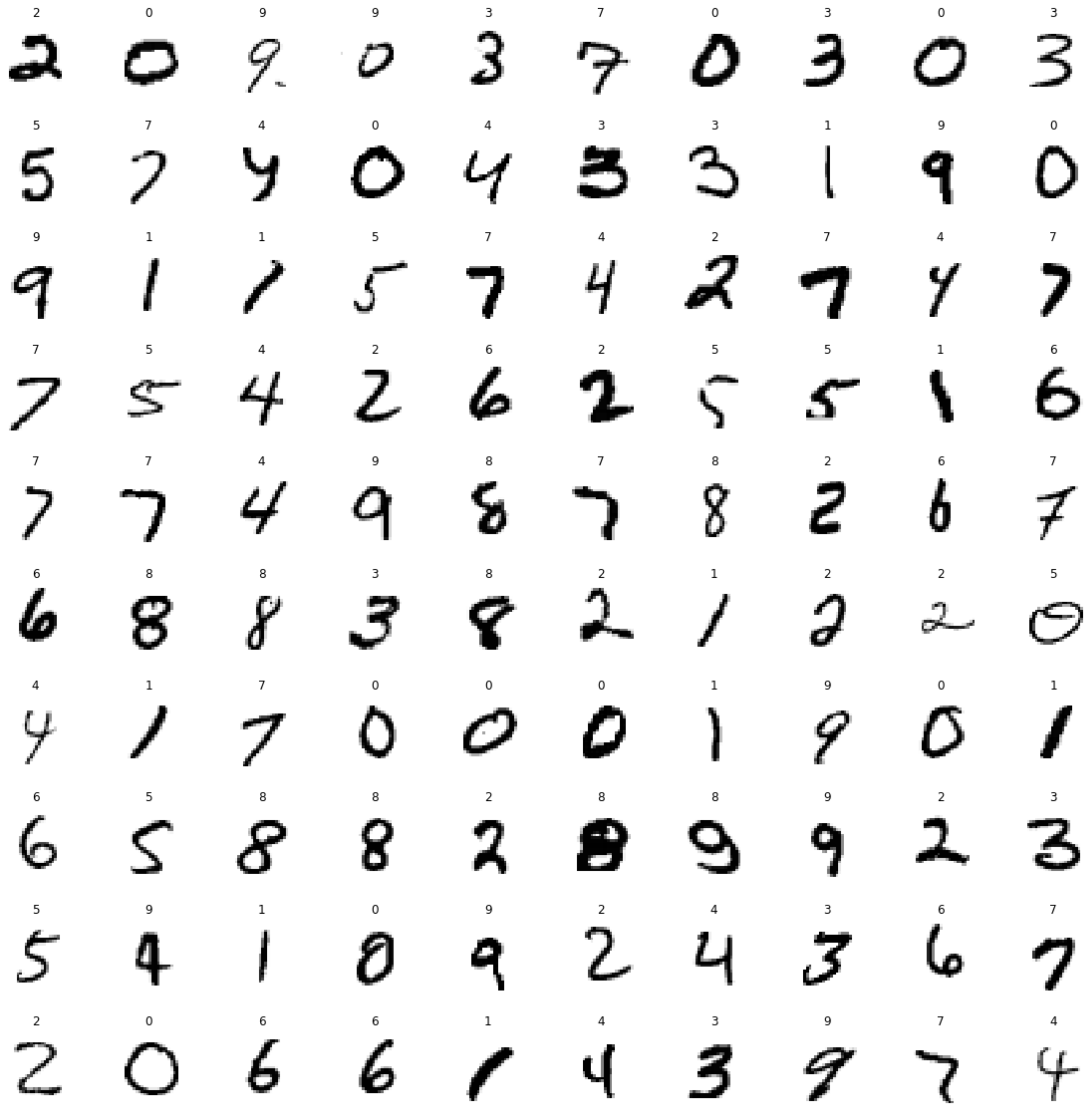
```
C:\Users\rocchm1\AppData\Roaming\Python\Python37\site-packages\tensorflow\python\keras\engine\sequ
ential.py:455: UserWarning: `model.predict_classes()` is deprecated and will be removed after 2021
-01-01. Please use instead:* `np.argmax(model.predict(x), axis=-1)`,   if your model does multi-cl
ass classification   (e.g. if it uses a `softmax` last-layer activation).* `(model.predict(x) > 0.
5).astype("int32")`,   if your model does binary classification   (e.g. if it uses a `sigmoid` las
t-layer activation).
  warnings.warn('`model.predict_classes()` is deprecated and '
WARNING:tensorflow:Model was constructed with shape (None, 28, 28) for input KerasTensor(type_spec
=TensorSpec(shape=(None, 28, 28), dtype=tf.float32, name='flatten_input'), name='flatten_input', d
escription="created by layer 'flatten_input'"), but it was called on an input with incompatible sh
ape (32, 784).
```

Out[100…

```
array([2, 0, 9, ..., 3, 9, 2], dtype=int64)
```

In [101…

```
plt.figure(figsize=(20,20))
for index, image in enumerate(X_test[:100].reshape(100, 28, 28)):
    plt.subplot(10, 10, index + 1)
    plt.imshow(image, cmap="binary", interpolation="nearest")
    plt.axis('off')
    plt.title(y_pred[index], fontsize=12)
plt.subplots_adjust(wspace=0.2, hspace=0.5)
plt.show()
```



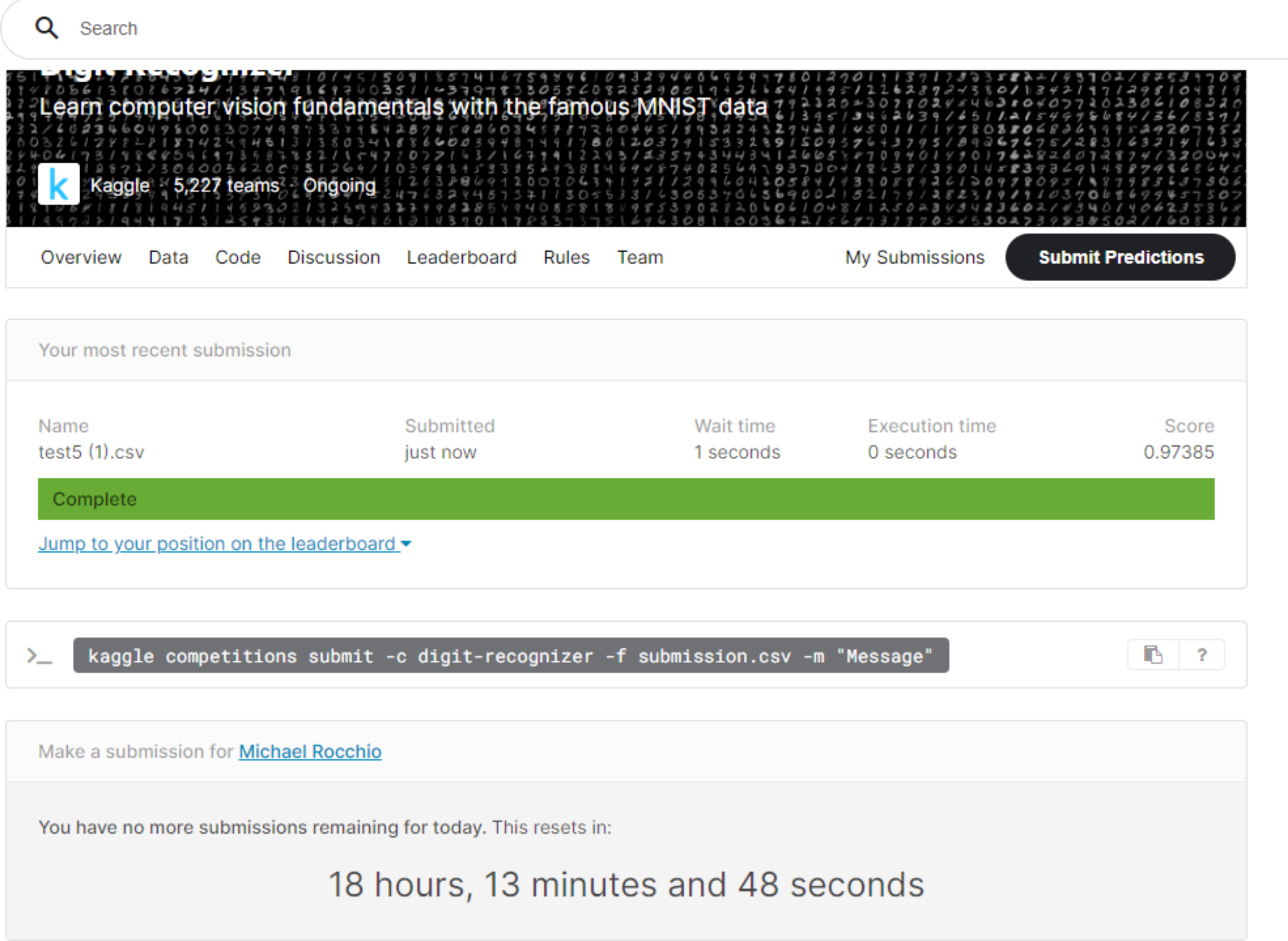
In [104…

```
sample_sub['Label']=y_pred
sample_sub.to_csv('test5.csv', index=False)
```

In [105…

```
from IPython.display import Image
Image(filename='test5.png')
######
######  MY USERNAME IN KAGGLE IS michaelrocchio ######
######
```

Out[105…



Learn computer vision fundamentals with the famous MNIST data

Your most recent submission

Name	Submitted	Wait time	Execution time	Score
test5 (1).csv	just now	1 seconds	0 seconds	0.97385

Complete

Jump to your position on the leaderboard

```
kaggle competitions submit -c digit-recognizer -f submission.csv -m "Message"
```

Make a submission for Michael Rocchio

You have no more submissions remaining for today. This resets in:

18 hours, 13 minutes and 48 seconds