In [1]:

```
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.metrics import confusion_matrix
```

In [2]:

```
mnist = tf.keras.datasets.mnist
(X_train_full, y_train_full), (X_test, y_test) = mnist.load_data()
```

In [3]:

```
X_valid = X_train_full[:5000] / 255.0
X_train = X_train_full[5000:] / 255.0
X_test = X_test / 255.0

y_valid = y_train_full[:5000]
y_train = y_train_full[5000:]

X_train = X_train[..., np.newaxis]
X_valid = X_valid[..., np.newaxis]
X_test = X_test[..., np.newaxis]
```

In [13]:

```
from functools import partial

my_dense_layer = partial(tf.keras.layers.Dense, activation="relu", kernel_regularizer=tf.keras.regu
larizers.l2(0.0001))
my_conv_layer = partial(tf.keras.layers.Conv2D, activation="tanh", padding="valid")

model = tf.keras.models.Sequential([
    my_conv_layer(6,5,padding="same",input_shape=[28,28,1]),
    tf.keras.layers.MaxPooling2D(2),
    my_conv_layer(16,5),
    tf.keras.layers.MaxPooling2D(2),
    my_conv_layer(120,5),
    tf.keras.layers.Flatten(),
    my_dense_layer(84),
    my_dense_layer(10, activation="softmax")
])
```

In [14]:

```
model.compile(loss="sparse_categorical_crossentropy",
              optimizer=tf.keras.optimizers.Adam(learning_rate=0.001),
              metrics=["accuracy"])
```

In [15]:

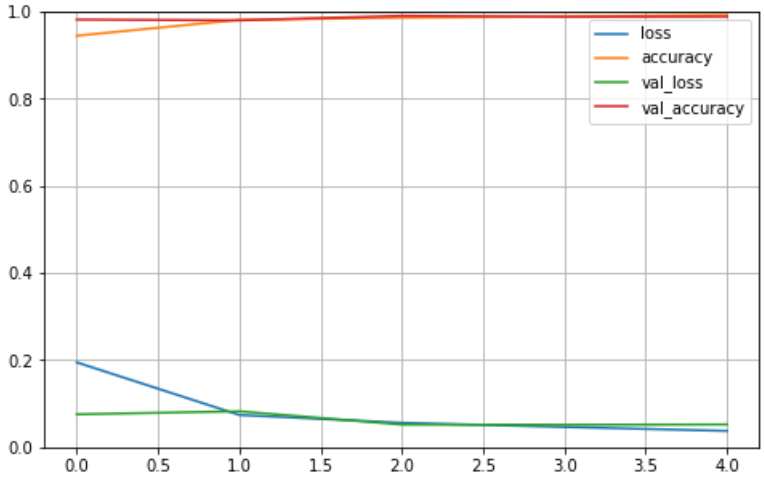
```
history = model.fit(X_train, y_train, epochs=5, validation_data=(X_valid,y_valid))
```

```
Train on 55000 samples, validate on 5000 samples
Epoch 1/5
55000/55000 [==============================] - 13s 230us/sample - loss: 0.1948 - accuracy: 0.9447
- val_loss: 0.0756 - val_accuracy: 0.9818
Epoch 2/5
55000/55000 [==============================] - 13s 228us/sample - loss: 0.0740 - accuracy: 0.9811
- val_loss: 0.0824 - val_accuracy: 0.9800
Epoch 3/5
55000/55000 [==============================] - 12s 227us/sample - loss: 0.0560 - accuracy: 0.9862
- val_loss: 0.0519 - val_accuracy: 0.9904
Epoch 4/5
55000/55000 [==============================] - 13s 230us/sample - loss: 0.0464 - accuracy: 0.9887
- val_loss: 0.0514 - val_accuracy: 0.9880
Epoch 5/5
```

```
Epoch 5/5
55000/55000 [==============================] - 13s 232us/sample - loss: 0.0374 - accuracy: 0.9922
- val_loss: 0.0523 - val_accuracy: 0.9886
```

In [16]:

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



In [17]:

```
y_pred = model.predict_classes(X_train)
conf_train = confusion_matrix(y_train, y_pred)
print(conf_train)
```

```
[[5427    0    4    0    0    0   10    0    3    0]
 [   1 6110    6    0    5    1    3   35   18    0]
 [   0    0 5458    0    1    0    0    8    3    0]
 [   0    1    9 5603    0    8    0   10    3    4]
 [   2    0    6    0 5264    0   25    1    2    7]
 [   1    0    0    8    1 4953   19    0    3    2]
 [   1    1    1    0    0    0 5411    0    3    0]
 [   1    2    4    0    6    0    0 5694    0    8]
 [   0    1   10    3    0    2    2    3 5367    1]
 [  10    0    2    2   38    2    0   11   13 5376]]
```

In [18]:

```
model.evaluate(X_test,y_test)
```

```
10000/10000 [==============================] - 1s 89us/sample - loss: 0.0457 - accuracy: 0.9884
```

Out[18]:

```
[0.045718724122643474, 0.9884]
```

In [19]:

```
y_pred = model.predict_classes(X_test)
conf_test = confusion_matrix(y_test, y_pred)
print(conf_test)
```

```
[[ 972    0    2    0    0    0    2    1    3    0]
 [   0 1121    1    0    0    0    2    7    4    0]
 [   1    0 1026    0    0    0    0    5    0    0]
 [   0    0    3 1000    0    3    0    2    2    0]
 [   1    0    0    0  974    0    4    2    0    1]
 [   2    0    0   10    0  877    2    1    0    0]
 [   3    1    1    0    1    2  948    0    2    0]
 [   1    0    3    0    0    0    0 1020    1    3]
```

```
[   2    0    5    2    1    0    1    3  960    0]
[   3    0    0    4   10    2    0    2    2  986]]
```

In [20]:

```
fig, ax = plt.subplots()

# hide axes
fig.patch.set_visible(False)
ax.axis('off')
ax.axis('tight')

# create table and save to file
df = pd.DataFrame(conf_test)
ax.table(cellText=df.values, rowLabels=np.arange(10), colLabels=np.arange(10), loc='center', cellLo
c='center')
fig.tight_layout()
plt.savefig('conf_mat_2.png')
```

	0	1	2	3	4	5	6	7	8	9
0	972	0	2	0	0	0	2	1	3	0
1	0	1121	1	0	0	0	2	7	4	0
2	1	0	1026	0	0	0	0	5	0	0
3	0	0	3	1000	0	3	0	2	2	0
4	1	0	0	0	974	0	4	2	0	1
5	2	0	0	10	0	877	2	1	0	0
6	3	1	1	0	1	2	948	0	2	0
7	1	0	3	0	0	0	0	1020	1	3
8	2	0	5	2	1	0	1	3	960	0
9	3	0	0	4	10	2	0	2	2	986

In [ ]: