Project 1 - CNN code

This notebook contains all code used for CNN model design and testing. The data are preserved for possible future use. Due to the time cost for training the CNN model, all cells are remained from the testing. Therfore, the cell running index will not be in order.

In [275]:
```
# Set system error to null for keras backend message
import os
import sys
stderr = sys.stderr
sys.stderr = open(os.devnull, 'w')
```

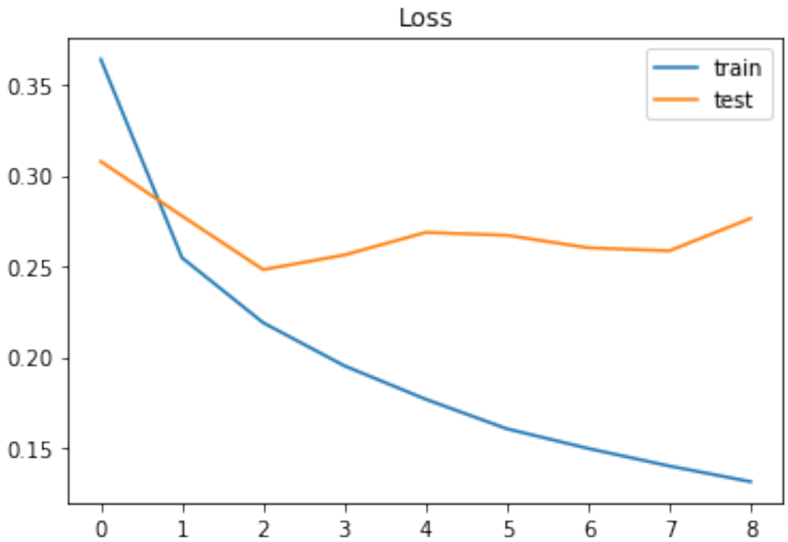
In [1]:
```
import numpy as np
from sklearn.model_selection import train_test_split
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.datasets.fashion_mnist import load_data

# Load FMNIST data from tensorflow data base
(X_train, y_train), (X_test, y_test) = load_data()

# Split the training data into validation data using 9:1 ratio
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, te
st_size=0.1, shuffle=True, random_state=1)

# Preprocess the images
X_train = X_train.astype('float32')/255.0
X_val = X_val.astype('float32')/255.0
X_test = X_test.astype('float32')/255.0

X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], X_t
rain.shape[2], 1))
X_val = np.reshape(X_val, (X_val.shape[0], X_val.shape[1], X_val.shape
[2], 1))
X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], X_test.
shape[2], 1))
```

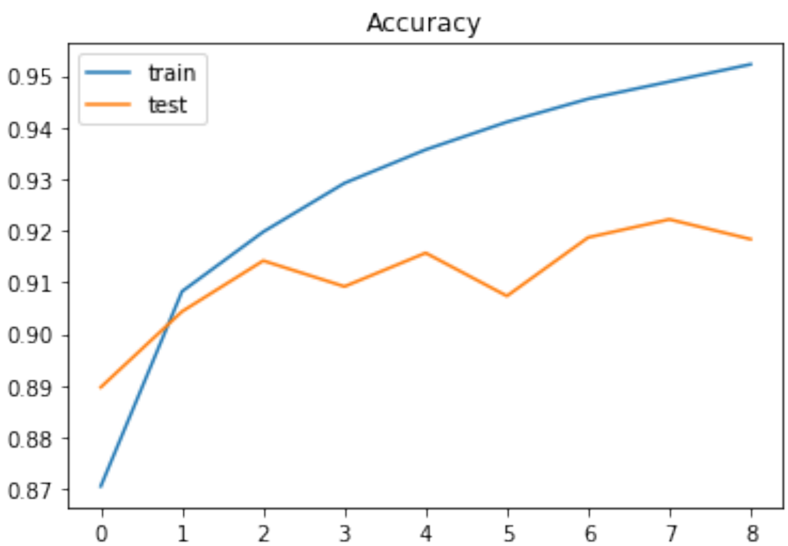
```
(60000, 28, 28) (60000,) (10000, 28, 28) (10000,)
```

In [234]:

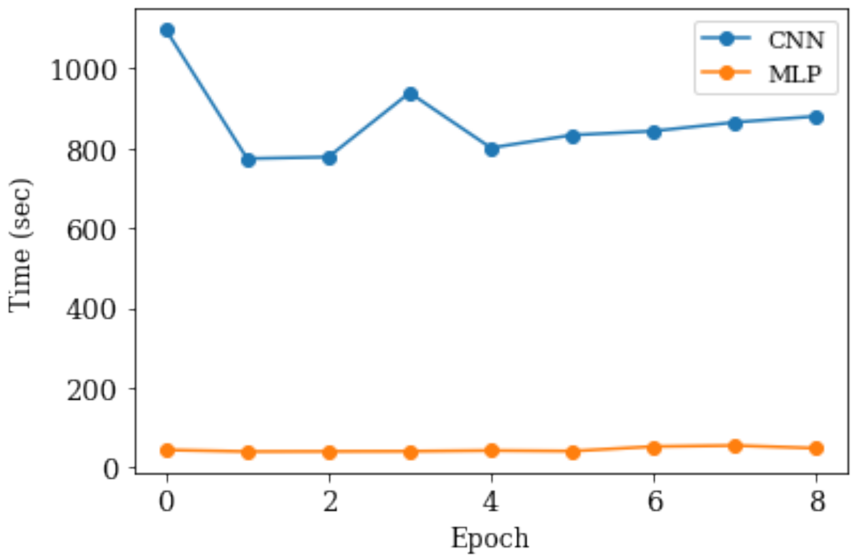
```
# Build the model
from tensorflow import keras
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Dense, Dropout, Flatten
, AveragePooling2D
from keras.optimizers import Adam, SGD
from keras.callbacks import EarlyStopping
from matplotlib import pyplot

model = Sequential()

# Add convolution layer adn pooling layer
model.add(Conv2D(128, (3, 3), activation='relu', input_shape=(28, 28,
1)))
model.add(Conv2D(256, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

# Droup unnecessary data to avoid overfitting
model.add(Dropout(0.25))
model.add(Flatten())

# Sort classes
model.add(Dense(10, activation='softmax'))

adam = Adam(lr=0.001)
model.compile(optimizer=adam, loss='sparse_categorical_crossentropy',
                      metrics=['accuracy'])

# Determine early stopping criteria
es = EarlyStopping(monitor='val_accuracy', mode='max', patience=3)
history = model.fit(X_train, y_train, validation_data=(X_val, y_val),
                            epochs=9, batch_size=5, callbacks=[es])

# Plot the performance for each epoch
pyplot.figure(1)
pyplot.plot(history.history['loss'], label='train')
pyplot.plot(history.history['val_loss'], label='test')
pyplot.title('Loss')
pyplot.legend()

pyplot.figure(2)
pyplot.plot(history.history['accuracy'], label='train')
pyplot.plot(history.history['val_accuracy'], label='test')
pyplot.title('Accuracy')
pyplot.legend()
pyplot.show()
```

```
Train on 54000 samples, validate on 6000 samples
Epoch 1/9
54000/54000 [==============================] - 1097s 20ms/step - los
s: 0.3639 - accuracy: 0.8705 - val_loss: 0.3077 - val_accuracy: 0.88
97
Epoch 2/9
54000/54000 [==============================] - 774s 14ms/step - loss
: 0.2547 - accuracy: 0.9082 - val_loss: 0.2777 - val_accuracy: 0.904
3
Epoch 3/9
54000/54000 [==============================] - 778s 14ms/step - loss
: 0.2189 - accuracy: 0.9198 - val_loss: 0.2481 - val_accuracy: 0.914
2
Epoch 4/9
54000/54000 [==============================] - 939s 17ms/step - loss
: 0.1952 - accuracy: 0.9292 - val_loss: 0.2562 - val_accuracy: 0.909
2
Epoch 5/9
54000/54000 [==============================] - 801s 15ms/step - loss
: 0.1768 - accuracy: 0.9357 - val_loss: 0.2686 - val_accuracy: 0.915
7
Epoch 6/9
54000/54000 [==============================] - 833s 15ms/step - loss
: 0.1606 - accuracy: 0.9410 - val_loss: 0.2670 - val_accuracy: 0.907
3
Epoch 7/9
54000/54000 [==============================] - 843s 16ms/step - loss
: 0.1497 - accuracy: 0.9455 - val_loss: 0.2602 - val_accuracy: 0.918
7
Epoch 8/9
54000/54000 [==============================] - 865s 16ms/step - loss
: 0.1401 - accuracy: 0.9488 - val_loss: 0.2585 - val_accuracy: 0.922
2
Epoch 9/9
54000/54000 [==============================] - 880s 16ms/step - loss
: 0.1315 - accuracy: 0.9522 - val_loss: 0.2763 - val_accuracy: 0.918
3
```


Loss
train
test


Accuracy
train
test

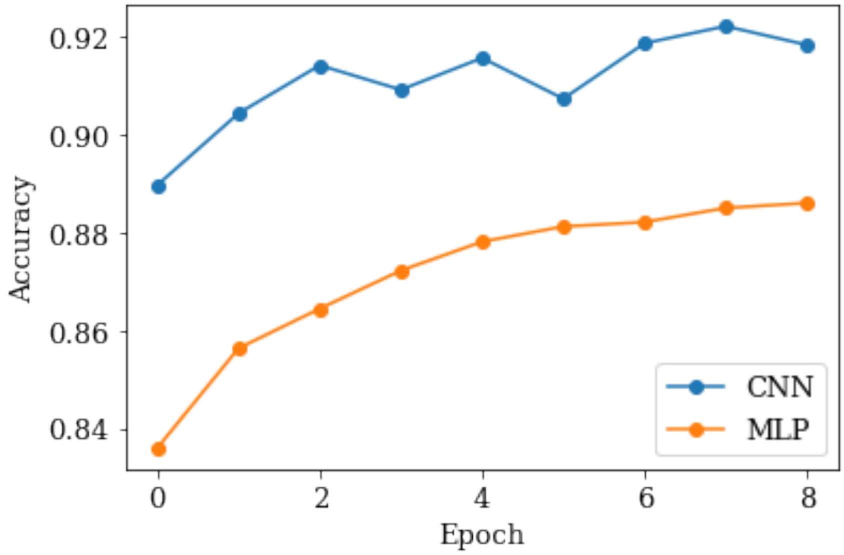
In [271]:
```
time1 = [1097, 774, 778, 939, 801, 833, 843, 865, 880]
time2 = [44.31, 40.12, 40.52, 40.90, 42.68, 41.29, 52.78, 55.33, 48.58
]
nepochs = [0, 1, 2, 3, 4, 5, 6, 7, 8]
fig44, ax44 = pyplot.subplots()
ax44.plot(time1, 'o-', label='CNN')
ax44.plot(time2, 'o-', label='MLP')
ax44.set_xlabel('Epoch', fontsize=12)
ax44.set_ylabel('Time (sec)', fontsize=12)
ax44.legend(loc=1, prop={'size': 11})
pyplot.rcParams.update({'font.size': 13})
pyplot.show()
fig44.savefig('cnn_mlp_time.pdf')
```



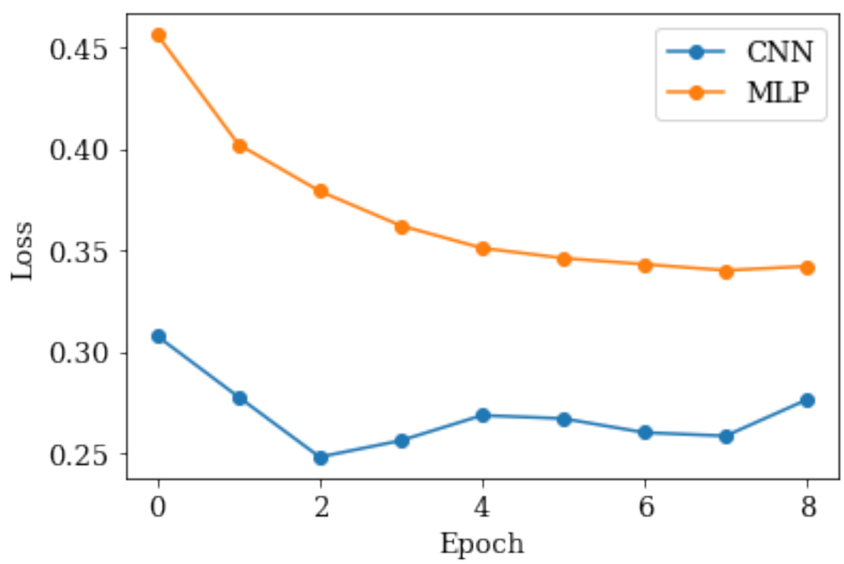
In [245]:
```
best_accu = history.history['accuracy']
best_loss = history.history['loss']
best_test_accu = history.history['val_accuracy']
best_test_loss = history.history['val_loss']
```

In [261]:
```
mlp_accu = [0.8359, 0.8563, 0.8644, 0.8722, 0.8781, 0.8812, 0.8821, 0.
8850, 0.8860]
mlp_loss = [0.456, 0.402, 0.379, 0.362, 0.351, 0.346, 0.343, 0.340, 0.
342]
```

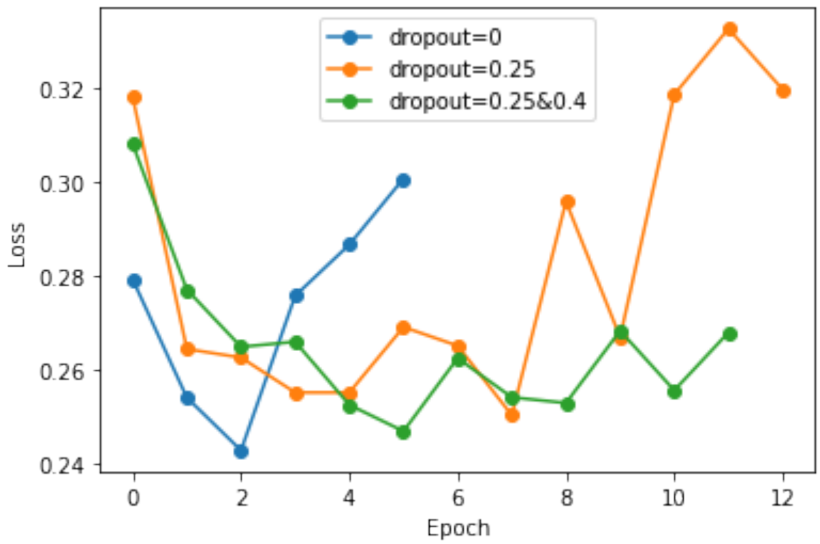
In [272]:
```
fig55, ax55 = pyplot.subplots()
ax55.plot(best_test_accu, 'o-', label='CNN')
ax55.plot(mlp_accu, 'o-', label='MLP')
ax55.set_xlabel('Epoch')
ax55.set_ylabel('Accuracy')
ax55.legend()
pyplot.show()
fig55.savefig('cnn_mlp_accuracy.pdf')
```



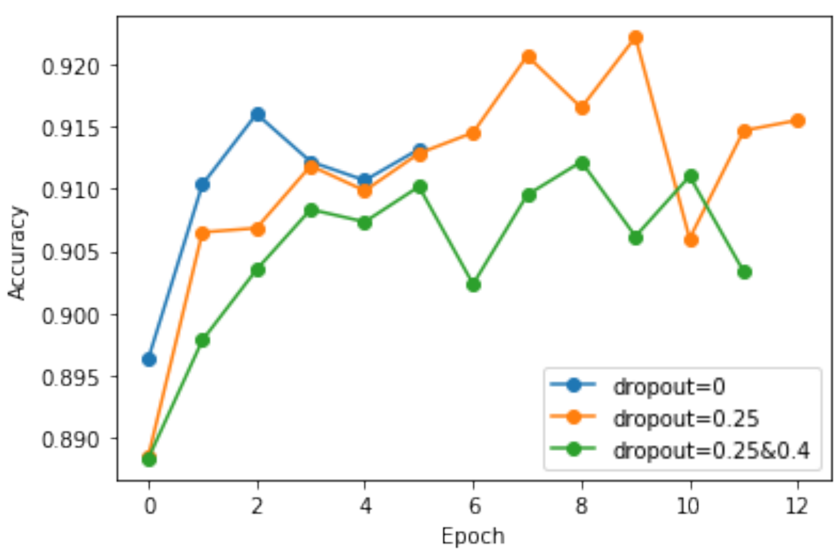
In [273]:
```
fig66, ax66 = pyplot.subplots()
ax66.plot(best_test_loss, 'o-', label='CNN')
ax66.plot(mlp_loss, 'o-', label='MLP')
ax66.set_xlabel('Epoch')
ax66.set_ylabel('Loss')
ax66.legend()
pyplot.show()
fig66.savefig('cnn_mlp_loss.pdf')
```



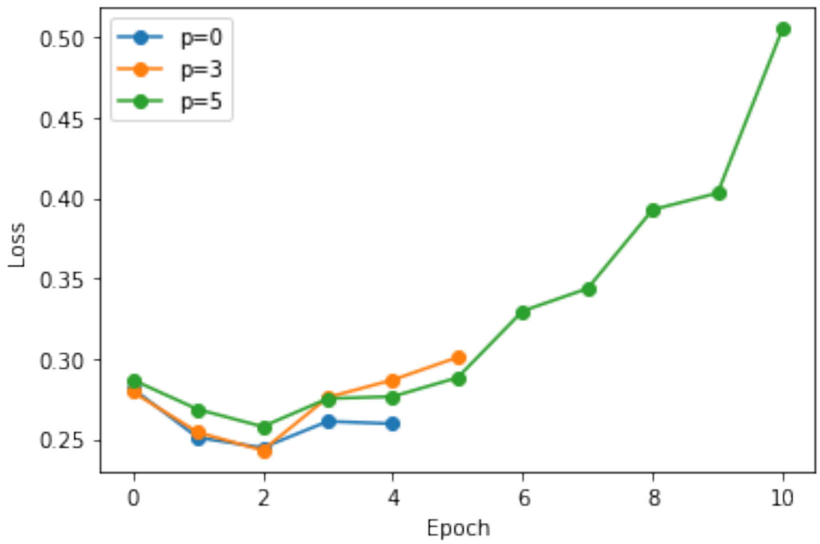
In [230]:
```
fig25, ax25 = pyplot.subplots()
ax25.plot(p3_test_loss, 'o-', label='dropout=0')
ax25.plot(p3d25_test_loss, 'o-', label='dropout=0.25')
ax25.plot(p3d25d4_test_loss, 'o-', label='dropout=0.25&0.4')
ax25.set_xlabel('Epoch')
ax25.set_ylabel('Loss')
ax25.legend()
pyplot.show()
fig25.savefig('cnn_dloss.pdf')
```



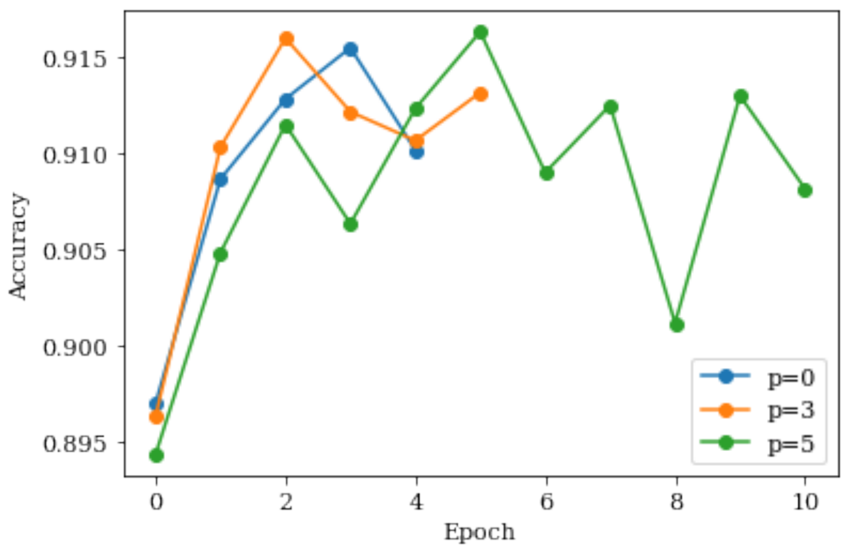
In [231]:
```
fig26, ax26 = pyplot.subplots()
ax26.plot(p3_test_accu, 'o-', label='dropout=0')
ax26.plot(p3d25_test_accu, 'o-', label='dropout=0.25')
ax26.plot(p3d25d4_test_accu, 'o-', label='dropout=0.25&0.4')
ax26.set_xlabel('Epoch')
ax26.set_ylabel('Accuracy')
ax26.legend()
pyplot.show()
fig26.savefig('cnn_daccuracy.pdf')
```



In [232]:
```
fig23, ax23 = pyplot.subplots()
ax23.plot(p0_test_loss, 'o-', label='p=0')
ax23.plot(p3_test_loss, 'o-', label='p=3')
ax23.plot(p5_test_loss, 'o-', label='p=5')
ax23.set_xlabel('Epoch')
ax23.set_ylabel('Loss')
ax23.legend()
pyplot.show()
fig23.savefig('cnn_es_loss.pdf')
```



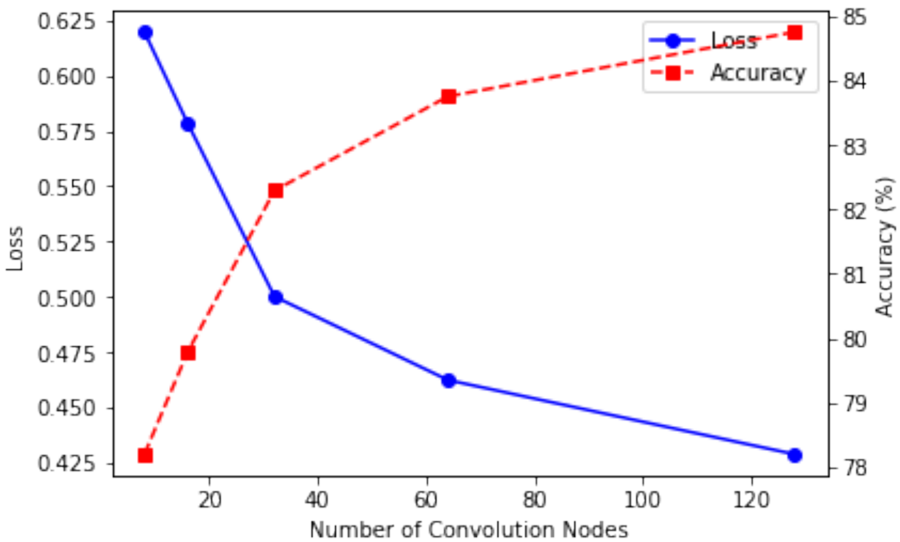
In [287]:
```
fig24, ax24 = pyplot.subplots()
ax24.plot(p0_test_accu, 'o-', label='p=0')
ax24.plot(p3_test_accu, 'o-', label='p=3')
ax24.plot(p5_test_accu, 'o-', label='p=5')
ax24.set_xlabel('Epoch')
ax24.set_ylabel('Accuracy')

ax24.legend()
pyplot.show()
fig24.savefig('cnn_es_accuracy.pdf')
```



In [129]:
```
# Test results for hyperparameters

# Batch size test
batch_size = [5, 10, 20, 32, 64, 128]
batch_loss2 = [0.3596, 0.3639, 0.3742, 0.3856, 0.4176, 0.4716]
batch_accu2 = [0.8729*100, 0.8692*100, 0.8669*100, 0.8626*100, 0.8519*
100, 0.8334*100]

# Learning rate
np = [0.1, 0.01, 0.001, 0.0001, 0.00001]
np_loss = [2.9452, 0.4868, 0.3595, 0.4590, 0.7533]
np_accu = [0.0972*100, 0.8245*100, 0.8714*100, 0.8363*100, 0.7429*100]

# Convolution nodes
nconv = [8, 16, 32, 64, 128]
nconv_loss = [0.6198, 0.5784, 0.5001, 0.4623, 0.4287]
nconv_accu = [0.7820*100, 0.7979*100, 0.8229*100, 0.8375*100, 0.8475*1
00]

# Convolution layer settings
slayer_loss = 0.3657
slayer_accu = 0.8714
dlayer_21_loss = 0.3620
dlayer_21_accu = 0.8699
dlyaer_22_loss = 0.3533
dlayer_22_accu = 0.8733
dlayer_12_loss = 0.3520
dlayer_12_accu = 0.8739

# Convolution kernel size
ck5_loss = 0.3839
ck5_accu = 0.8629
ck3_loss = 0.3520
ck3_accu = 0.8739

# Max pooling and average pooling
mp_loss = 0.3520
mp_accu = 0.8739
ap_loss = 0.3653
ap_accu = 0.8702

# Optimizer selection
adam_loss = 0.3653
adam_accu = 0.8702
sgd_loss = 0.7479
sgd_accu = 0.7368

# Loss function
centro_loss = 0.3587
centro_accu = 0.8724
scentro_loss = 0.3509
scentro_accu = 0.8748
```

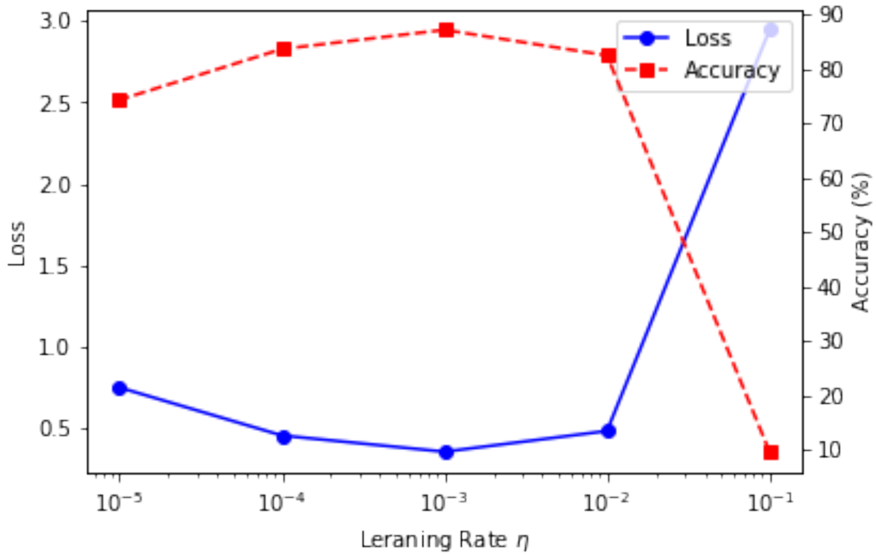
In [134]:

```
fig, ax5 = pyplot.subplots()
l5 = ax5.plot(nconv, nconv_loss, 'bo-', label='Loss')
ax5.set_xlabel('Number of Convolution Nodes')
ax5.set_ylabel('Loss')
ax6 = ax5.twinx()
l6 = ax6.plot(nconv, nconv_accu, 'rs--', label='Accuracy')
ax6.set_ylabel('Accuracy (%)')
leg = l5+l6
labs = [l.get_label() for l in leg]
ax5.legend(leg, labs, loc='upper right')
pyplot.show()
fig.savefig('num_conv.pdf')
```



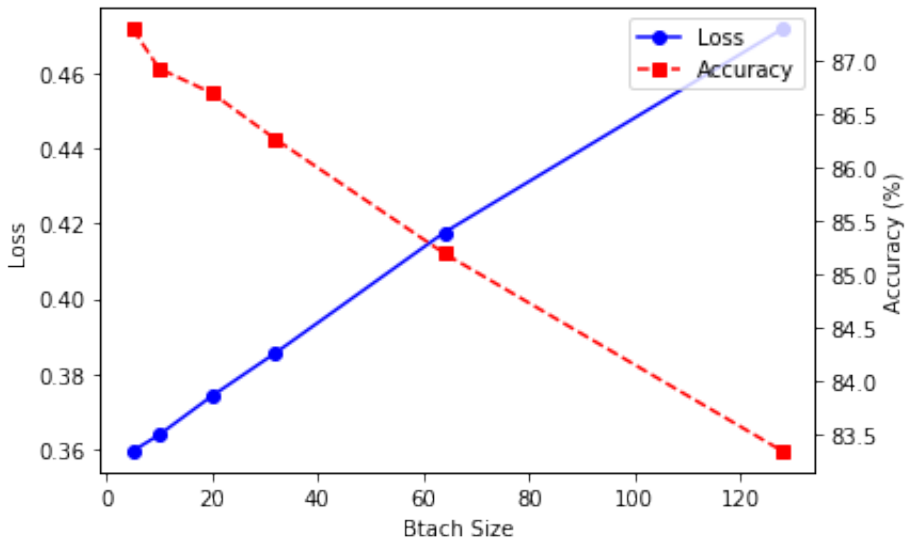
In [187]:

```
fig, ax3 = pyplot.subplots()
l3 = ax3.plot(np, np_loss, 'bo-', label='Loss')
ax3.set_xscale('log')
ax3.set_xlabel('Leraning Rate $\eta$')
ax3.set_ylabel('Loss')
ax4 = ax3.twinx()
l4 = ax4.plot(np, np_accu, 'rs--', label='Accuracy')
ax4.set_ylabel('Accuracy (%)')
leg = l3+l4
labs = [l.get_label() for l in leg]
ax3.legend(leg, labs, loc='upper right')
pyplot.show()
fig.savefig('learning_rate.pdf')
```



In [186]:

```
fig, ax1 = pyplot.subplots()
l1 = ax1.plot(batch_size, batch_loss2, 'bo-', label='Loss')
ax1.set_xlabel('Btach Size')
ax1.set_ylabel('Loss')
ax2 = ax1.twinx()
l2 = ax2.plot(batch_size, batch_accu2, 'rs--', label='Accuracy')
ax2.set_ylabel('Accuracy (%)')
leg = l1+l2
labs = [l.get_label() for l in leg]
ax1.legend(leg, labs, loc='upper right')
pyplot.show()
fig.savefig('batch_size.pdf')
```



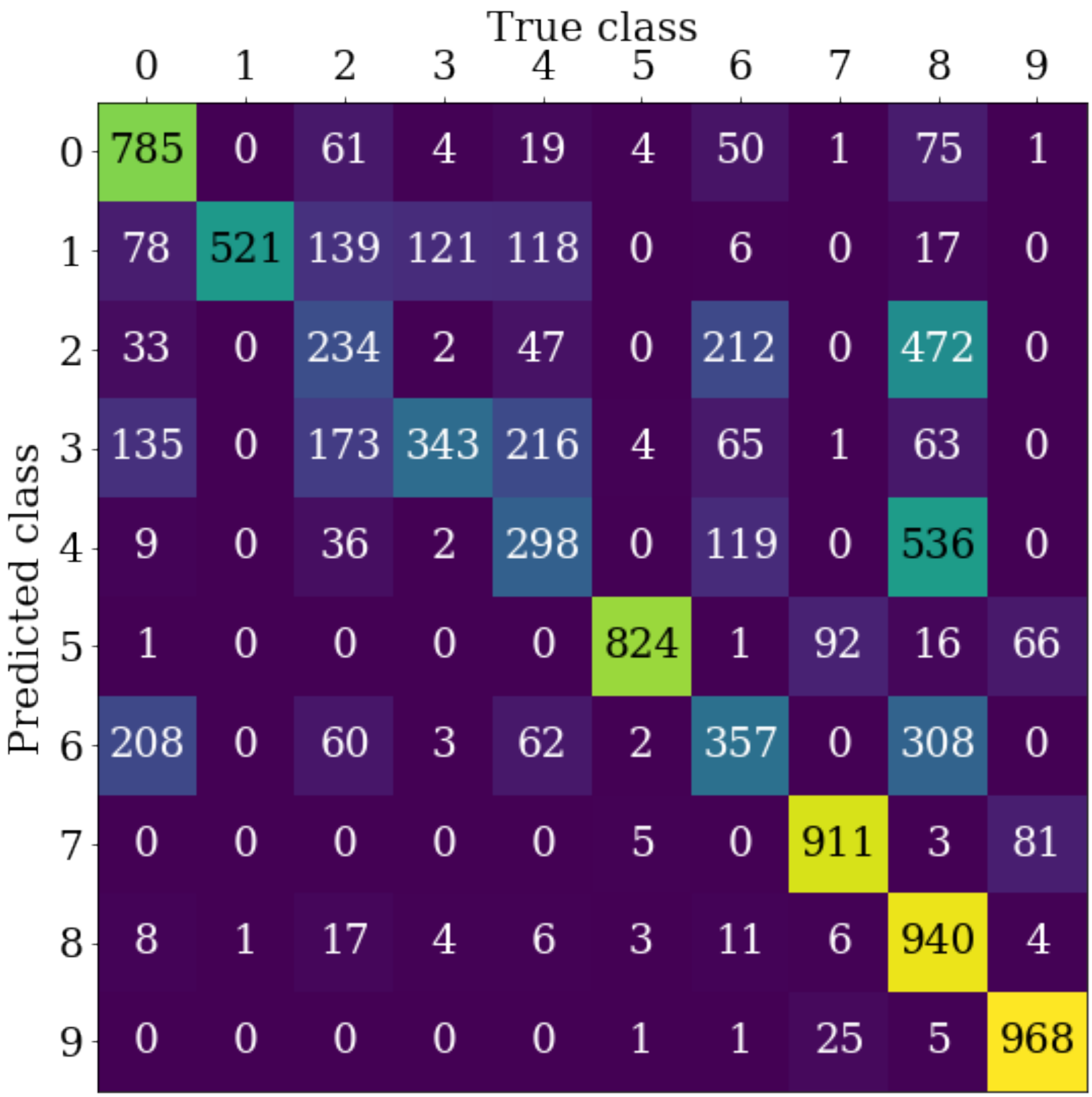
In [ ]:

```
def plot_conf(conf, tosave=None):
# Plot the confusion matrix

    import matplotlib
    import matplotlib.pyplot as plt
    matplotlib.rc('font', size=20, family='serif')
    fig, ax = plt.subplots(figsize=(8, 8))
    ax.imshow(conf)
    color_threshold = conf.max() / 2.
    for x in range(conf.shape[1]):
        for y in range(conf.shape[0]):
            val = conf[y, x]
            color = 'w' if val < color_threshold else 'k'
            ax.text(x, y, '%d'%val, ha='center', va='center', color=co
lor, fontsize=20)
    ax.set_xticks(np.arange(conf.shape[1]))
    ax.set_yticks(np.arange(conf.shape[0]))
    ax.set_xticklabels(np.arange(conf.shape[1]))
    ax.set_yticklabels(np.arange(conf.shape[0]))
    ax.set_xlabel('True class')
    ax.set_ylabel('Predicted class')
    ax.xaxis.tick_top()
    ax.xaxis.set_label_position('top')
    ax.set_aspect('equal')
    fig.tight_layout()
    if type(tosave)!=type(None): fig.savefig(tosave); print('Saved: %s
'%tosave)
    plt.show()
```

In [274]:

```
from sklearn.metrics import confusion_matrix

# Evalated the model with testing set
score = model.evaluate(X_test, y_test)
print('Test loss:', score[0])
print('Test accuracy', score[1])

# Reload data for confusion matrix calculation
(train, train_label), (test, test_label) = load_data()
test = test.reshape(10000, 28, 28, 1)
pred = model.predict_classes(test)

# Generate confusion matrix and plot it
conf = confusion_matrix(test_label, pred)
conf_tosave = 'cnn_conf.pdf'
plot_conf(conf, conf_tosave)
```

```
10000/10000 [==============================] - 44s 4ms/step
Test loss: 0.2914980470627546
Test accuracy 0.9139000177383423
Saved: cnn_conf.pdf
```




```
In [ ]:
```