# Assignment 12 - Neural Networks image recognition

Use both MLNN and the ConvNet to solve the following problem.

1. Add random noise (i.e. `np.random.normal` ) to the images in training and testing. Make sure each image gets a different noise feature added to it. Inspect by printing out an image.
2. Compare the loss/accuracy (train, val) after N epochs for both MLNN and ConvNet with and without noise.
3. Vary the amount of noise (multiply `np.random.normal` by a factor) and keep track of the accuracy and loss (for training and validation) and plot these results.

# Neural Networks - Image Recognition

In [1]:

```
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.optimizers import RMSprop
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras import backend
```

```
Using TensorFlow backend.
```

In [2]:

```
import matplotlib.pyplot as  plt
%matplotlib inline
```

## Multi Layer Neural Network

Trains a simple deep NN on the MNIST dataset. Gets to 98.40% test accuracy after 20 epochs (there is *a lot* of margin for parameter tuning).

In [3]:

```
# the data, shuffled and split between train and test sets
(x_train, y_train), (x_test, y_test) = mnist.load_data()

x_train = x_train.reshape(60000, 784)
x_test = x_test.reshape(10000, 784)
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255
print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples')
```

```
60000 train samples
10000 test samples
```

In [4]:

```
import numpy as np

x_train_noise10 = x_train + np.random.normal(0, 255*.10, x_train.shape)
x_test_noise10 = x_test + np.random.normal(0, 255*.10, x_test.shape)

x_train_noise15 = x_train + np.random.normal(0, 255*.15, x_train.shape)
x_test_noise15 = x_test + np.random.normal(0, 255*.15, x_test.shape)

x_train_noise20 = x_train + np.random.normal(0, 255*.20, x_train.shape)
x_test_noise20 = x_test + np.random.normal(0, 255*.20, x_test.shape)

x_train_noise50 = x_train + np.random.normal(0, 255*.50, x_train.shape)
x_test_noise50 = x_test + np.random.normal(0, 255*.50, x_test.shape)
```

In [5]:

```python
batch_size = 128
num_classes = 10
epochs = 20

# convert class vectors to binary class matrices
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)

model = Sequential()
model.add(Dense(512, activation='relu', input_shape=(784,)))
model.add(Dropout(0.2))
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(10, activation='softmax'))

model.summary()

model.compile(loss='categorical_crossentropy',
              optimizer=RMSprop(),
              metrics=['accuracy'])

history = model.fit(x_train, y_train,
                    batch_size=batch_size,
                    epochs=epochs,
                    verbose=1,
                    validation_data=(x_test, y_test))
score_nn = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score_nn[0])
print('Test accuracy:', score_nn[1])
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                Output Shape              Param #
=================================================================
dense (Dense)               (None, 512)               401920

_________________________________________________________________
dropout (Dropout)           (None, 512)               0

_________________________________________________________________
dense_1 (Dense)             (None, 512)               262656

_________________________________________________________________
dropout_1 (Dropout)         (None, 512)               0

_________________________________________________________________
dense_2 (Dense)             (None, 10)                5130
=================================================================
Total params: 669,706
Trainable params: 669,706
Non-trainable params: 0
_________________________________________________________________
Epoch 1/20
469/469 [==============================] - 13s 28ms/step - loss: 0.2420 - accur
acy: 0.9265 - val_loss: 0.0950 - val_accuracy: 0.9700
Epoch 2/20
469/469 [==============================] - 11s 23ms/step - loss: 0.1022 - accur
acy: 0.9696 - val_loss: 0.0842 - val_accuracy: 0.9734
Epoch 3/20
469/469 [==============================] - 11s 23ms/step - loss: 0.0763 - accur
acy: 0.9772 - val_loss: 0.0742 - val_accuracy: 0.9783
```

```
Epoch 4/20
469/469 [==============================] - 10s 21ms/step - loss: 0.0611 - accur
acy: 0.9815 - val_loss: 0.1013 - val_accuracy: 0.9728
Epoch 5/20
469/469 [==============================] - 10s 21ms/step - loss: 0.0505 - accur
acy: 0.9850 - val_loss: 0.0745 - val_accuracy: 0.9796
Epoch 6/20
469/469 [==============================] - 11s 24ms/step - loss: 0.0442 - accur
acy: 0.9871 - val_loss: 0.0749 - val_accuracy: 0.9799
Epoch 7/20
469/469 [==============================] - 9s 20ms/step - loss: 0.0379 - accura
cy: 0.9884 - val_loss: 0.0862 - val_accuracy: 0.9811
Epoch 8/20
469/469 [==============================] - 9s 19ms/step - loss: 0.0327 - accura
cy: 0.9903 - val_loss: 0.0734 - val_accuracy: 0.9840
Epoch 9/20
469/469 [==============================] - 9s 19ms/step - loss: 0.0307 - accura
cy: 0.9911 - val_loss: 0.0797 - val_accuracy: 0.9830
Epoch 10/20
469/469 [==============================] - 9s 19ms/step - loss: 0.0296 - accura
cy: 0.9915 - val_loss: 0.0843 - val_accuracy: 0.9825
Epoch 11/20
469/469 [==============================] - 9s 19ms/step - loss: 0.0263 - accura
cy: 0.9924 - val_loss: 0.0900 - val_accuracy: 0.9843
Epoch 12/20
469/469 [==============================] - 9s 19ms/step - loss: 0.0249 - accura
cy: 0.9929 - val_loss: 0.0914 - val_accuracy: 0.9826
Epoch 13/20
469/469 [==============================] - 9s 19ms/step - loss: 0.0218 - accura
cy: 0.9939 - val_loss: 0.0984 - val_accuracy: 0.9816
Epoch 14/20
469/469 [==============================] - 9s 19ms/step - loss: 0.0213 - accura
cy: 0.9937 - val_loss: 0.1053 - val_accuracy: 0.9825
Epoch 15/20
469/469 [==============================] - 9s 19ms/step - loss: 0.0211 - accura
cy: 0.9940 - val_loss: 0.1096 - val_accuracy: 0.9827
Epoch 16/20
469/469 [==============================] - 9s 19ms/step - loss: 0.0207 - accura
cy: 0.9946 - val_loss: 0.1252 - val_accuracy: 0.9820
Epoch 17/20
469/469 [==============================] - 9s 19ms/step - loss: 0.0199 - accura
cy: 0.9947 - val_loss: 0.1305 - val_accuracy: 0.9818
Epoch 18/20
469/469 [==============================] - 9s 19ms/step - loss: 0.0193 - accura
cy: 0.9947 - val_loss: 0.1170 - val_accuracy: 0.9824
Epoch 19/20
469/469 [==============================] - 9s 18ms/step - loss: 0.0176 - accura
cy: 0.9954 - val_loss: 0.1086 - val_accuracy: 0.9836
Epoch 20/20
469/469 [==============================] - 9s 19ms/step - loss: 0.0171 - accura
cy: 0.9955 - val_loss: 0.1388 - val_accuracy: 0.9835
Test loss: 0.13880014419555664
Test accuracy: 0.9835000038146973
```

In [6]:

```
batch_size = 128
num_classes = 10
epochs = 20

model = Sequential()
model.add(Dense(512, activation='relu', input_shape=(784,)))
model.add(Dropout(0.2))
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(10, activation='softmax'))

model.summary()

model.compile(loss='categorical_crossentropy',
              optimizer=RMSprop(),
              metrics=['accuracy'])

history = model.fit(x_train_noise10, y_train,
                    batch_size=batch_size,
                    epochs=epochs,
                    verbose=1,
                    validation_data=(x_test_noise10, y_test))
score_nn10 = model.evaluate(x_test_noise10, y_test, verbose=0)
print('Test loss:', score_nn10[0])
print('Test accuracy:', score_nn10[1])
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_3 (Dense)              (None, 512)               401920    
_________________________________________________________________
dropout_2 (Dropout)          (None, 512)               0         
_________________________________________________________________
dense_4 (Dense)              (None, 512)               262656    
_________________________________________________________________
dropout_3 (Dropout)          (None, 512)               0         
_________________________________________________________________
dense_5 (Dense)              (None, 10)                5130      
=================================================================
Total params: 669,706
Trainable params: 669,706
Non-trainable params: 0
_________________________________________________________________
Epoch 1/20
469/469 [==============================] - 9s 19ms/step - loss: 3.8833 - accura
cy: 0.1090 - val_loss: 2.3022 - val_accuracy: 0.1145
Epoch 2/20
469/469 [==============================] - 9s 19ms/step - loss: 2.3060 - accura
cy: 0.1167 - val_loss: 2.3069 - val_accuracy: 0.1119
Epoch 3/20
469/469 [==============================] - 9s 18ms/step - loss: 2.2923 - accura
cy: 0.1225 - val_loss: 2.3038 - val_accuracy: 0.1136
Epoch 4/20
469/469 [==============================] - 9s 18ms/step - loss: 2.2827 - accura
cy: 0.1265 - val_loss: 2.3095 - val_accuracy: 0.1130
```

```
Epoch 5/20
469/469 [==============================] - 9s 19ms/step - loss: 2.2728 - accura
cy: 0.1303 - val_loss: 2.3025 - val_accuracy: 0.1132
Epoch 6/20
469/469 [==============================] - 9s 18ms/step - loss: 2.2694 - accura
cy: 0.1336 - val_loss: 2.3037 - val_accuracy: 0.1126
Epoch 7/20
469/469 [==============================] - 9s 19ms/step - loss: 2.2644 - accura
cy: 0.1360 - val_loss: 2.3047 - val_accuracy: 0.1134
Epoch 8/20
469/469 [==============================] - 9s 19ms/step - loss: 2.2627 - accura
cy: 0.1377 - val_loss: 2.3022 - val_accuracy: 0.1133
Epoch 9/20
469/469 [==============================] - 9s 19ms/step - loss: 2.2606 - accura
cy: 0.1379 - val_loss: 2.3027 - val_accuracy: 0.1137
Epoch 10/20
469/469 [==============================] - 9s 19ms/step - loss: 2.2593 - accura
cy: 0.1396 - val_loss: 2.3032 - val_accuracy: 0.1129
Epoch 11/20
469/469 [==============================] - 9s 19ms/step - loss: 2.2606 - accura
cy: 0.1408 - val_loss: 2.3054 - val_accuracy: 0.1129
Epoch 12/20
469/469 [==============================] - 9s 19ms/step - loss: 2.2539 - accura
cy: 0.1425 - val_loss: 2.3042 - val_accuracy: 0.1134
Epoch 13/20
469/469 [==============================] - 9s 19ms/step - loss: 2.2583 - accura
cy: 0.1423 - val_loss: 2.3047 - val_accuracy: 0.1136
Epoch 14/20
469/469 [==============================] - 9s 19ms/step - loss: 2.2525 - accura
cy: 0.1444 - val_loss: 2.3042 - val_accuracy: 0.1137
Epoch 15/20
469/469 [==============================] - 9s 19ms/step - loss: 2.2529 - accura
cy: 0.1440 - val_loss: 2.3048 - val_accuracy: 0.1133
Epoch 16/20
469/469 [==============================] - 9s 19ms/step - loss: 2.2508 - accura
cy: 0.1454 - val_loss: 2.3098 - val_accuracy: 0.1129
Epoch 17/20
469/469 [==============================] - 9s 19ms/step - loss: 2.2494 - accura
cy: 0.1456 - val_loss: 2.3095 - val_accuracy: 0.1131
Epoch 18/20
469/469 [==============================] - 9s 19ms/step - loss: 2.2513 - accura
cy: 0.1451 - val_loss: 2.3055 - val_accuracy: 0.1135
Epoch 19/20
469/469 [==============================] - 9s 19ms/step - loss: 2.2513 - accura
cy: 0.1472 - val_loss: 2.3069 - val_accuracy: 0.1131
Epoch 20/20
469/469 [==============================] - 9s 19ms/step - loss: 2.2538 - accura
cy: 0.1460 - val_loss: 2.3062 - val_accuracy: 0.1128
Test loss: 2.306216239929199
Test accuracy: 0.1128000020980835
```

In [7]:

```
batch_size = 128
num_classes = 10
epochs = 20

model = Sequential()
model.add(Dense(512, activation='relu', input_shape=(784,)))
model.add(Dropout(0.2))
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(10, activation='softmax'))

model.summary()

model.compile(loss='categorical_crossentropy',
              optimizer=RMSprop(),
              metrics=['accuracy'])

history = model.fit(x_train_noise15, y_train,
                    batch_size=batch_size,
                    epochs=epochs,
                    verbose=1,
                    validation_data=(x_test_noise15, y_test))
score_nn15 = model.evaluate(x_test_noise15, y_test, verbose=0)
print('Test loss:', score_nn15[0])
print('Test accuracy:', score_nn15[1])
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_6 (Dense)              (None, 512)               401920    
_________________________________________________________________
dropout_4 (Dropout)          (None, 512)               0         
_________________________________________________________________
dense_7 (Dense)              (None, 512)               262656    
_________________________________________________________________
dropout_5 (Dropout)          (None, 512)               0         
_________________________________________________________________
dense_8 (Dense)              (None, 10)                5130      
=================================================================
Total params: 669,706
Trainable params: 669,706
Non-trainable params: 0
_________________________________________________________________
Epoch 1/20
469/469 [==============================] - 9s 19ms/step - loss: 4.8214 - accura
cy: 0.1085 - val_loss: 2.3049 - val_accuracy: 0.1130
Epoch 2/20
469/469 [==============================] - 9s 18ms/step - loss: 2.3061 - accura
cy: 0.1163 - val_loss: 2.3020 - val_accuracy: 0.1138
Epoch 3/20
469/469 [==============================] - 9s 18ms/step - loss: 2.2963 - accura
cy: 0.1184 - val_loss: 2.3022 - val_accuracy: 0.1134
Epoch 4/20
469/469 [==============================] - 9s 18ms/step - loss: 2.2953 - accura
cy: 0.1213 - val_loss: 2.3032 - val_accuracy: 0.1133
```

```
Epoch 5/20
469/469 [==============================] - 9s 18ms/step - loss: 2.2874 - accura
cy: 0.1230 - val_loss: 2.3029 - val_accuracy: 0.1134
Epoch 6/20
469/469 [==============================] - 9s 18ms/step - loss: 2.2867 - accura
cy: 0.1239 - val_loss: 2.3024 - val_accuracy: 0.1138
Epoch 7/20
469/469 [==============================] - 9s 19ms/step - loss: 2.2862 - accura
cy: 0.1256 - val_loss: 2.3015 - val_accuracy: 0.1137
Epoch 8/20
469/469 [==============================] - 8s 18ms/step - loss: 2.2831 - accura
cy: 0.1268 - val_loss: 2.3043 - val_accuracy: 0.1134
Epoch 9/20
469/469 [==============================] - 9s 18ms/step - loss: 2.2812 - accura
cy: 0.1275 - val_loss: 2.3019 - val_accuracy: 0.1135
Epoch 10/20
469/469 [==============================] - 8s 18ms/step - loss: 2.2810 - accura
cy: 0.1287 - val_loss: 2.3032 - val_accuracy: 0.1133
Epoch 11/20
469/469 [==============================] - 8s 18ms/step - loss: 2.2791 - accura
cy: 0.1294 - val_loss: 2.3016 - val_accuracy: 0.1137
Epoch 12/20
469/469 [==============================] - 9s 18ms/step - loss: 2.2751 - accura
cy: 0.1304 - val_loss: 2.3023 - val_accuracy: 0.1135
Epoch 13/20
469/469 [==============================] - 8s 18ms/step - loss: 2.2756 - accura
cy: 0.1309 - val_loss: 2.3026 - val_accuracy: 0.1135
Epoch 14/20
469/469 [==============================] - 8s 18ms/step - loss: 2.2732 - accura
cy: 0.1319 - val_loss: 2.3038 - val_accuracy: 0.1135
Epoch 15/20
469/469 [==============================] - 8s 18ms/step - loss: 2.2714 - accura
cy: 0.1321 - val_loss: 2.3029 - val_accuracy: 0.1135
Epoch 16/20
469/469 [==============================] - 8s 18ms/step - loss: 2.2689 - accura
cy: 0.1328 - val_loss: 2.3053 - val_accuracy: 0.1133
Epoch 17/20
469/469 [==============================] - 9s 18ms/step - loss: 2.2706 - accura
cy: 0.1335 - val_loss: 2.3041 - val_accuracy: 0.1139
Epoch 18/20
469/469 [==============================] - 9s 18ms/step - loss: 2.2710 - accura
cy: 0.1338 - val_loss: 2.3035 - val_accuracy: 0.1132
Epoch 19/20
469/469 [==============================] - 8s 18ms/step - loss: 2.2679 - accura
cy: 0.1342 - val_loss: 2.3016 - val_accuracy: 0.1136
Epoch 20/20
469/469 [==============================] - 8s 18ms/step - loss: 2.2681 - accura
cy: 0.1345 - val_loss: 2.3017 - val_accuracy: 0.1135
Test loss: 2.301741123199463
Test accuracy: 0.11349999904632568
```

In [8]:

```
batch_size = 128
num_classes = 10
epochs = 20

model = Sequential()
model.add(Dense(512, activation='relu', input_shape=(784,)))
model.add(Dropout(0.2))
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(10, activation='softmax'))

model.summary()

model.compile(loss='categorical_crossentropy',
              optimizer=RMSprop(),
              metrics=['accuracy'])

history = model.fit(x_train_noise20, y_train,
                    batch_size=batch_size,
                    epochs=epochs,
                    verbose=1,
                    validation_data=(x_test_noise20, y_test))
score_nn20 = model.evaluate(x_test_noise20, y_test, verbose=0)
print('Test loss:', score_nn20[0])
print('Test accuracy:', score_nn20[1])
```

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_9 (Dense)              (None, 512)               401920    
_________________________________________________________________
dropout_6 (Dropout)          (None, 512)               0         
_________________________________________________________________
dense_10 (Dense)             (None, 512)               262656    
_________________________________________________________________
dropout_7 (Dropout)          (None, 512)               0         
_________________________________________________________________
dense_11 (Dense)             (None, 10)                5130      
=================================================================
Total params: 669,706
Trainable params: 669,706
Non-trainable params: 0
_________________________________________________________________
Epoch 1/20
469/469 [==============================] - 9s 19ms/step - loss: 5.9826 - accura
cy: 0.1077 - val_loss: 2.3026 - val_accuracy: 0.1138
Epoch 2/20
469/469 [==============================] - 9s 18ms/step - loss: 2.3088 - accura
cy: 0.1151 - val_loss: 2.3059 - val_accuracy: 0.1133
Epoch 3/20
469/469 [==============================] - 9s 19ms/step - loss: 2.3020 - accura
cy: 0.1174 - val_loss: 2.3051 - val_accuracy: 0.1135
Epoch 4/20
469/469 [==============================] - 9s 18ms/step - loss: 2.2957 - accura
cy: 0.1189 - val_loss: 2.3022 - val_accuracy: 0.1134
```

```
Epoch 5/20
469/469 [==============================] - 9s 18ms/step - loss: 2.2964 - accura
cy: 0.1201 - val_loss: 2.3021 - val_accuracy: 0.1140
Epoch 6/20
469/469 [==============================] - 9s 20ms/step - loss: 2.2946 - accura
cy: 0.1204 - val_loss: 2.3022 - val_accuracy: 0.1135
Epoch 7/20
469/469 [==============================] - 9s 18ms/step - loss: 2.2886 - accura
cy: 0.1218 - val_loss: 2.3018 - val_accuracy: 0.1134
Epoch 8/20
469/469 [==============================] - 9s 18ms/step - loss: 2.2908 - accura
cy: 0.1222 - val_loss: 2.3019 - val_accuracy: 0.1133
Epoch 9/20
469/469 [==============================] - 9s 19ms/step - loss: 2.2884 - accura
cy: 0.1230 - val_loss: 2.3023 - val_accuracy: 0.1135
Epoch 10/20
469/469 [==============================] - 9s 18ms/step - loss: 2.2875 - accura
cy: 0.1238 - val_loss: 2.3013 - val_accuracy: 0.1137
Epoch 11/20
469/469 [==============================] - 9s 18ms/step - loss: 2.2865 - accura
cy: 0.1241 - val_loss: 2.3025 - val_accuracy: 0.1135
Epoch 12/20
469/469 [==============================] - 9s 18ms/step - loss: 2.2856 - accura
cy: 0.1248 - val_loss: 2.3017 - val_accuracy: 0.1136
Epoch 13/20
469/469 [==============================] - 9s 18ms/step - loss: 2.2829 - accura
cy: 0.1252 - val_loss: 2.3021 - val_accuracy: 0.1137
Epoch 14/20
469/469 [==============================] - 9s 18ms/step - loss: 2.2838 - accura
cy: 0.1257 - val_loss: 2.3019 - val_accuracy: 0.1137
Epoch 15/20
469/469 [==============================] - 9s 19ms/step - loss: 2.2806 - accura
cy: 0.1262 - val_loss: 2.3017 - val_accuracy: 0.1134
Epoch 16/20
469/469 [==============================] - 9s 19ms/step - loss: 2.2808 - accura
cy: 0.1265 - val_loss: 2.3040 - val_accuracy: 0.1133
Epoch 17/20
469/469 [==============================] - 9s 19ms/step - loss: 2.2826 - accura
cy: 0.1268 - val_loss: 2.3057 - val_accuracy: 0.1135
Epoch 18/20
469/469 [==============================] - 9s 19ms/step - loss: 2.2791 - accura
cy: 0.1270 - val_loss: 2.3025 - val_accuracy: 0.1137
Epoch 19/20
469/469 [==============================] - 9s 18ms/step - loss: 2.2760 - accura
cy: 0.1279 - val_loss: 2.3046 - val_accuracy: 0.1135
Epoch 20/20
469/469 [==============================] - 9s 19ms/step - loss: 2.2789 - accura
cy: 0.1276 - val_loss: 2.3024 - val_accuracy: 0.1138
Test loss: 2.3023934364318848
Test accuracy: 0.11379999667406082
```

In [9]:

```
batch_size = 128
num_classes = 10
epochs = 20

model = Sequential()
model.add(Dense(512, activation='relu', input_shape=(784,)))
model.add(Dropout(0.2))
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(10, activation='softmax'))

model.summary()

model.compile(loss='categorical_crossentropy',
              optimizer=RMSprop(),
              metrics=['accuracy'])

history = model.fit(x_train_noise50, y_train,
                    batch_size=batch_size,
                    epochs=epochs,
                    verbose=1,
                    validation_data=(x_test_noise50, y_test))
score_nn50 = model.evaluate(x_test_noise50, y_test, verbose=0)
print('Test loss:', score_nn50[0])
print('Test accuracy:', score_nn50[1])
```

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_12 (Dense)             (None, 512)               401920    
_________________________________________________________________
dropout_8 (Dropout)          (None, 512)               0         
_________________________________________________________________
dense_13 (Dense)             (None, 512)               262656    
_________________________________________________________________
dropout_9 (Dropout)          (None, 512)               0         
_________________________________________________________________
dense_14 (Dense)             (None, 10)                5130      
=================================================================
Total params: 669,706
Trainable params: 669,706
Non-trainable params: 0
_________________________________________________________________
Epoch 1/20
469/469 [==============================] - 9s 19ms/step - loss: 10.7068 - accur
acy: 0.1089 - val_loss: 2.3028 - val_accuracy: 0.1133
Epoch 2/20
469/469 [==============================] - 9s 18ms/step - loss: 2.3158 - accura
cy: 0.1141 - val_loss: 2.3021 - val_accuracy: 0.1135
Epoch 3/20
469/469 [==============================] - 9s 18ms/step - loss: 2.3138 - accura
cy: 0.1149 - val_loss: 2.3019 - val_accuracy: 0.1135
Epoch 4/20
469/469 [==============================] - 8s 18ms/step - loss: 2.3077 - accura
cy: 0.1157 - val_loss: 2.3014 - val_accuracy: 0.1135
Epoch 5/20
```

```
469/469 [==============================] - 8s 18ms/step - loss: 2.3072 - accura
cy: 0.1160 - val_loss: 2.3011 - val_accuracy: 0.1135
Epoch 6/20
469/469 [==============================] - 9s 18ms/step - loss: 2.3066 - accura
cy: 0.1163 - val_loss: 2.3015 - val_accuracy: 0.1134
Epoch 7/20
469/469 [==============================] - 9s 18ms/step - loss: 2.3064 - accura
cy: 0.1164 - val_loss: 2.3014 - val_accuracy: 0.1135
Epoch 8/20
469/469 [==============================] - 9s 19ms/step - loss: 2.3032 - accura
cy: 0.1172 - val_loss: 2.3021 - val_accuracy: 0.1134
Epoch 9/20
469/469 [==============================] - 9s 18ms/step - loss: 2.3020 - accura
cy: 0.1172 - val_loss: 2.3012 - val_accuracy: 0.1135
Epoch 10/20
469/469 [==============================] - 9s 19ms/step - loss: 2.3021 - accura
cy: 0.1170 - val_loss: 2.3007 - val_accuracy: 0.1137
Epoch 11/20
469/469 [==============================] - 9s 19ms/step - loss: 2.2985 - accura
cy: 0.1173 - val_loss: 2.3015 - val_accuracy: 0.1135
Epoch 12/20
469/469 [==============================] - 9s 19ms/step - loss: 2.3007 - accura
cy: 0.1176 - val_loss: 2.3016 - val_accuracy: 0.1134
Epoch 13/20
469/469 [==============================] - 9s 18ms/step - loss: 2.2982 - accura
cy: 0.1178 - val_loss: 2.3010 - val_accuracy: 0.1136
Epoch 14/20
469/469 [==============================] - 9s 18ms/step - loss: 2.3017 - accura
cy: 0.1177 - val_loss: 2.3007 - val_accuracy: 0.1138
Epoch 15/20
469/469 [==============================] - 9s 18ms/step - loss: 2.2959 - accura
cy: 0.1181 - val_loss: 2.3009 - val_accuracy: 0.1135
Epoch 16/20
469/469 [==============================] - 9s 19ms/step - loss: 2.2938 - accura
cy: 0.1180 - val_loss: 2.3016 - val_accuracy: 0.1134
Epoch 17/20
469/469 [==============================] - 9s 18ms/step - loss: 2.2947 - accura
cy: 0.1179 - val_loss: 2.3012 - val_accuracy: 0.1136
Epoch 18/20
469/469 [==============================] - 8s 18ms/step - loss: 2.2973 - accura
cy: 0.1185 - val_loss: 2.3015 - val_accuracy: 0.1135- l
Epoch 19/20
469/469 [==============================] - 8s 18ms/step - loss: 2.2946 - accura
cy: 0.1186 - val_loss: 2.3010 - val_accuracy: 0.1135
Epoch 20/20
469/469 [==============================] - 9s 19ms/step - loss: 2.2968 - accura
cy: 0.1183 - val_loss: 2.3011 - val_accuracy: 0.1135
Test loss: 2.3010566234588623
Test accuracy: 0.11349999904632568
```

## Conv Net

Trains a simple convnet on the MNIST dataset. Gets to 99.25% test accuracy after 12 epochs (there is still a lot of margin for parameter tuning).

In [10]:

```
# input image dimensions
img_rows, img_cols = 28, 28

# the data, shuffled and split between train and test sets
(x_train, y_train), (x_test, y_test) = mnist.load_data()

if backend.image_data_format() == 'channels_first':
    x_train = x_train.reshape(x_train.shape[0], 1, img_rows, img_cols)
    x_test = x_test.reshape(x_test.shape[0], 1, img_rows, img_cols)
    input_shape = (1, img_rows, img_cols)
else:
    x_train = x_train.reshape(x_train.shape[0], img_rows, img_cols, 1)
    x_test = x_test.reshape(x_test.shape[0], img_rows, img_cols, 1)
    input_shape = (img_rows, img_cols, 1)

x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255
print('x_train shape:', x_train.shape)
print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples')
```

```
x_train shape: (60000, 28, 28, 1)
60000 train samples
10000 test samples
```

In [11]:

```
batch_size = 128
num_classes = 10
epochs = 12

# convert class vectors to binary class matrices
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)

model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3),
                 activation='relu',
                 input_shape=input_shape))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))

model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adadelta(),
              metrics=['accuracy'])

model.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(x_test, y_test))
score_cn = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score_cn[0])
print('Test accuracy:', score_cn[1])
```

```
Epoch 1/12
469/469 [==============================] - 117s 249ms/step - loss: 2.2752 - acc
uracy: 0.1705 - val_loss: 2.2320 - val_accuracy: 0.3669
Epoch 2/12
469/469 [==============================] - 115s 246ms/step - loss: 2.2066 - acc
uracy: 0.2875 - val_loss: 2.1504 - val_accuracy: 0.4971
Epoch 3/12
469/469 [==============================] - 116s 248ms/step - loss: 2.1202 - acc
uracy: 0.3871 - val_loss: 2.0392 - val_accuracy: 0.6126
Epoch 4/12
469/469 [==============================] - 117s 249ms/step - loss: 2.0047 - acc
uracy: 0.4634 - val_loss: 1.8926 - val_accuracy: 0.6838
Epoch 5/12
469/469 [==============================] - 117s 250ms/step - loss: 1.8616 - acc
uracy: 0.5206 - val_loss: 1.7147 - val_accuracy: 0.7254
Epoch 6/12
469/469 [==============================] - 117s 249ms/step - loss: 1.6990 - acc
uracy: 0.5627 - val_loss: 1.5176 - val_accuracy: 0.7540
Epoch 7/12
469/469 [==============================] - 116s 248ms/step - loss: 1.5319 - acc
uracy: 0.5975 - val_loss: 1.3216 - val_accuracy: 0.7777
Epoch 8/12
469/469 [==============================] - 117s 250ms/step - loss: 1.3794 - acc
```

```
uracy: 0.6245 - val_loss: 1.1469 - val_accuracy: 0.7989
Epoch 9/12
469/469 [==============================] - 116s 248ms/step - loss: 1.2448 - acc
uracy: 0.6525 - val_loss: 1.0017 - val_accuracy: 0.8154
Epoch 10/12
469/469 [==============================] - 115s 246ms/step - loss: 1.1344 - acc
uracy: 0.6745 - val_loss: 0.8867 - val_accuracy: 0.8284
Epoch 11/12
469/469 [==============================] - 117s 250ms/step - loss: 1.0516 - acc
uracy: 0.6927 - val_loss: 0.7986 - val_accuracy: 0.8375
Epoch 12/12
469/469 [==============================] - 117s 249ms/step - loss: 0.9770 - acc
uracy: 0.7114 - val_loss: 0.7276 - val_accuracy: 0.8449
Test loss: 0.7276210188865662
Test accuracy: 0.8449000120162964
```

In [12]:

```
x_train_noise10 = x_train + np.random.normal(0, 255*.10, x_train.shape)
x_test_noise10 = x_test + np.random.normal(0, 255*.10, x_test.shape)

x_train_noise15 = x_train + np.random.normal(0, 255*.15, x_train.shape)
x_test_noise15 = x_test + np.random.normal(0, 255*.15, x_test.shape)

x_train_noise20 = x_train + np.random.normal(0, 255*.20, x_train.shape)
x_test_noise20 = x_test + np.random.normal(0, 255*.20, x_test.shape)

x_train_noise50 = x_train + np.random.normal(0, 255*.50, x_train.shape)
x_test_noise50 = x_test + np.random.normal(0, 255*.50, x_test.shape)
```

In [13]:

```
batch_size = 128
num_classes = 10
epochs = 12

# convert class vectors to binary class matrices
# y_train = keras.utils.to_categorical(y_train, num_classes)
# y_test = keras.utils.to_categorical(y_test, num_classes)

model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3),
                 activation='relu',
                 input_shape=input_shape))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))

model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adadelta(),
              metrics=['accuracy'])

model.fit(x_train_noise10, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(x_test_noise10, y_test))
score_cn10 = model.evaluate(x_test_noise10, y_test, verbose=0)
print('Test loss:', score_cn10[0])
print('Test accuracy:', score_cn10[1])
```

```
Epoch 1/12
469/469 [==============================] - 118s 251ms/step - loss: 7.1525 - a
ccuracy: 0.1005 - val_loss: 2.4515 - val_accuracy: 0.1001
Epoch 2/12
469/469 [==============================] - 118s 253ms/step - loss: 2.8377 - a
ccuracy: 0.0994 - val_loss: 2.3031 - val_accuracy: 0.1003
Epoch 3/12
469/469 [==============================] - 118s 252ms/step - loss: 2.3872 - a
ccuracy: 0.1020 - val_loss: 2.3025 - val_accuracy: 0.1008
Epoch 4/12
469/469 [==============================] - 118s 252ms/step - loss: 2.3355 - a
ccuracy: 0.1011 - val_loss: 2.3025 - val_accuracy: 0.1010
Epoch 5/12
469/469 [==============================] - 122s 260ms/step - loss: 2.3204 - a
ccuracy: 0.1009 - val_loss: 2.3025 - val_accuracy: 0.1012
Epoch 6/12
469/469 [==============================] - 118s 251ms/step - loss: 2.3155 - a
ccuracy: 0.1072 - val_loss: 2.3025 - val_accuracy: 0.1136
Epoch 7/12
469/469 [==============================] - 118s 251ms/step - loss: 2.3111 - a
ccuracy: 0.1111 - val_loss: 2.3025 - val_accuracy: 0.1135
Epoch 8/12
469/469 [==============================] - 117s 249ms/step - loss: 2.3099 - a
```

```
ccuracy: 0.1116 - val_loss: 2.3025 - val_accuracy: 0.1135
Epoch 9/12
469/469 [==============================] - 117s 249ms/step - loss: 2.3088 - a
ccuracy: 0.1114 - val_loss: 2.3025 - val_accuracy: 0.1135
Epoch 10/12
469/469 [==============================] - 117s 249ms/step - loss: 2.3077 - a
ccuracy: 0.1120 - val_loss: 2.3025 - val_accuracy: 0.1135
Epoch 11/12
469/469 [==============================] - 117s 250ms/step - loss: 2.3068 - a
ccuracy: 0.1125 - val_loss: 2.3025 - val_accuracy: 0.1135
Epoch 12/12
469/469 [==============================] - 119s 254ms/step - loss: 2.3066 - a
ccuracy: 0.1120 - val_loss: 2.3025 - val_accuracy: 0.1135
Test loss: 2.302523612976074
Test accuracy: 0.11349999904632568
```

In [14]:

```
batch_size = 128
num_classes = 10
epochs = 12

# convert class vectors to binary class matrices
# y_train = keras.utils.to_categorical(y_train, num_classes)
# y_test = keras.utils.to_categorical(y_test, num_classes)

model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3),
                 activation='relu',
                 input_shape=input_shape))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))

model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adadelta(),
              metrics=['accuracy'])

model.fit(x_train_noise15, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(x_test_noise15, y_test))
score_cn15 = model.evaluate(x_test_noise15, y_test, verbose=0)
print('Test loss:', score_cn15[0])
print('Test accuracy:', score_cn15[1])
```

```
Epoch 1/12
469/469 [==============================] - 121s 259ms/step - loss: 9.6275 - a
ccuracy: 0.0999 - val_loss: 2.6429 - val_accuracy: 0.0973
Epoch 2/12
469/469 [==============================] - 126s 269ms/step - loss: 3.3043 - a
ccuracy: 0.0988 - val_loss: 2.3052 - val_accuracy: 0.0986
Epoch 3/12
469/469 [==============================] - 126s 269ms/step - loss: 2.4603 - a
ccuracy: 0.0970 - val_loss: 2.3026 - val_accuracy: 0.1141
Epoch 4/12
469/469 [==============================] - 122s 260ms/step - loss: 2.3572 - a
ccuracy: 0.1084 - val_loss: 2.3026 - val_accuracy: 0.1135
Epoch 5/12
469/469 [==============================] - 120s 256ms/step - loss: 2.3325 - a
ccuracy: 0.1104 - val_loss: 2.3025 - val_accuracy: 0.1135
Epoch 6/12
469/469 [==============================] - 120s 255ms/step - loss: 2.3234 - a
ccuracy: 0.1095 - val_loss: 2.3025 - val_accuracy: 0.1135
Epoch 7/12
469/469 [==============================] - 131s 279ms/step - loss: 2.3180 - a
ccuracy: 0.1106 - val_loss: 2.3025 - val_accuracy: 0.1135
Epoch 8/12
469/469 [==============================] - 122s 261ms/step - loss: 2.3152 - a
ccuracy: 0.1111 - val_loss: 2.3025 - val_accuracy: 0.1135
```

```
Epoch 9/12
469/469 [==============================] - 119s 254ms/step - loss: 2.3120 - a
ccuracy: 0.1116 - val_loss: 2.3025 - val_accuracy: 0.1135
Epoch 10/12
469/469 [==============================] - 118s 252ms/step - loss: 2.3106 - a
ccuracy: 0.1123 - val_loss: 2.3025 - val_accuracy: 0.1135
Epoch 11/12
469/469 [==============================] - 118s 252ms/step - loss: 2.3085 - a
ccuracy: 0.1122 - val_loss: 2.3025 - val_accuracy: 0.1135
Epoch 12/12
469/469 [==============================] - 118s 251ms/step - loss: 2.3089 - a
ccuracy: 0.1115 - val_loss: 2.3025 - val_accuracy: 0.1135
Test loss: 2.3025200366973877
Test accuracy: 0.11349999904632568
```

In [15]:

```
batch_size = 128
num_classes = 10
epochs = 12

# convert class vectors to binary class matrices
# y_train = keras.utils.to_categorical(y_train, num_classes)
# y_test = keras.utils.to_categorical(y_test, num_classes)

model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3),
                 activation='relu',
                 input_shape=input_shape))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))

model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adadelta(),
              metrics=['accuracy'])

model.fit(x_train_noise20, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(x_test_noise20, y_test))
score_cn20 = model.evaluate(x_test_noise20, y_test, verbose=0)
print('Test loss:', score_cn20[0])
print('Test accuracy:', score_cn20[1])
```

```
Epoch 1/12
469/469 [==============================] - 119s 253ms/step - loss: 14.6517 -
accuracy: 0.1010 - val_loss: 3.0254 - val_accuracy: 0.1003
Epoch 2/12
469/469 [==============================] - 120s 256ms/step - loss: 3.9864 - a
ccuracy: 0.0999 - val_loss: 2.3054 - val_accuracy: 0.1042
Epoch 3/12
469/469 [==============================] - 121s 258ms/step - loss: 2.5208 - a
ccuracy: 0.1022 - val_loss: 2.3026 - val_accuracy: 0.1023
Epoch 4/12
469/469 [==============================] - 120s 255ms/step - loss: 2.3815 - a
ccuracy: 0.1032 - val_loss: 2.3026 - val_accuracy: 0.1026
Epoch 5/12
469/469 [==============================] - 120s 255ms/step - loss: 2.3432 - a
ccuracy: 0.1046 - val_loss: 2.3026 - val_accuracy: 0.1026
Epoch 6/12
469/469 [==============================] - 136s 290ms/step - loss: 2.3284 - a
ccuracy: 0.1036 - val_loss: 2.3026 - val_accuracy: 0.1026
Epoch 7/12
469/469 [==============================] - 128s 272ms/step - loss: 2.3200 - a
ccuracy: 0.1041 - val_loss: 2.3026 - val_accuracy: 0.1027
Epoch 8/12
469/469 [==============================] - 133s 284ms/step - loss: 2.3191 - a
ccuracy: 0.1038 - val_loss: 2.3026 - val_accuracy: 0.1027
```

```
Epoch 9/12
469/469 [==============================] - 117s 249ms/step - loss: 2.3160 - a
ccuracy: 0.1042 - val_loss: 2.3025 - val_accuracy: 0.1027
Epoch 10/12
469/469 [==============================] - 116s 247ms/step - loss: 2.3134 - a
ccuracy: 0.1041 - val_loss: 2.3025 - val_accuracy: 0.1135
Epoch 11/12
469/469 [==============================] - 119s 253ms/step - loss: 2.3119 - a
ccuracy: 0.1116 - val_loss: 2.3025 - val_accuracy: 0.1135
Epoch 12/12
469/469 [==============================] - 117s 249ms/step - loss: 2.3105 - a
ccuracy: 0.1117 - val_loss: 2.3025 - val_accuracy: 0.1135
Test loss: 2.3025314807891846
Test accuracy: 0.11349999904632568
```

In [16]:

```
batch_size = 128
num_classes = 10
epochs = 12

# convert class vectors to binary class matrices
# y_train = keras.utils.to_categorical(y_train, num_classes)
# y_test = keras.utils.to_categorical(y_test, num_classes)

model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3),
                 activation='relu',
                 input_shape=input_shape))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))

model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adadelta(),
              metrics=['accuracy'])

model.fit(x_train_noise50, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(x_test_noise50, y_test))
score_cn50 = model.evaluate(x_test_noise50, y_test, verbose=0)
print('Test loss:', score_cn50[0])
print('Test accuracy:', score_cn50[1])
```

```
Epoch 1/12
469/469 [==============================] - 117s 249ms/step - loss: 33.0727 -
accuracy: 0.1005 - val_loss: 4.2993 - val_accuracy: 0.0938
Epoch 2/12
469/469 [==============================] - 117s 249ms/step - loss: 6.8540 - a
ccuracy: 0.1007 - val_loss: 2.3068 - val_accuracy: 0.1009
Epoch 3/12
469/469 [==============================] - 116s 246ms/step - loss: 2.8969 - a
ccuracy: 0.1011 - val_loss: 2.3028 - val_accuracy: 0.1135
Epoch 4/12
469/469 [==============================] - 115s 245ms/step - loss: 2.4970 - a
ccuracy: 0.1098 - val_loss: 2.3025 - val_accuracy: 0.1136
Epoch 5/12
469/469 [==============================] - 116s 248ms/step - loss: 2.4137 - a
ccuracy: 0.1113 - val_loss: 2.3026 - val_accuracy: 0.1135
Epoch 6/12
469/469 [==============================] - 115s 246ms/step - loss: 2.3721 - a
ccuracy: 0.1111 - val_loss: 2.3026 - val_accuracy: 0.1135
Epoch 7/12
469/469 [==============================] - 116s 248ms/step - loss: 2.3493 - a
ccuracy: 0.1119 - val_loss: 2.3026 - val_accuracy: 0.1135
Epoch 8/12
469/469 [==============================] - 115s 246ms/step - loss: 2.3419 - a
ccuracy: 0.1115 - val_loss: 2.3025 - val_accuracy: 0.1135
```

```
Epoch 9/12
469/469 [==============================] - 115s 245ms/step - loss: 2.3364 - a
ccuracy: 0.1122 - val_loss: 2.3025 - val_accuracy: 0.1135
Epoch 10/12
469/469 [==============================] - 116s 247ms/step - loss: 2.3261 - a
ccuracy: 0.1119 - val_loss: 2.3025 - val_accuracy: 0.1135
Epoch 11/12
469/469 [==============================] - 115s 246ms/step - loss: 2.3237 - a
ccuracy: 0.1124 - val_loss: 2.3025 - val_accuracy: 0.1135
Epoch 12/12
469/469 [==============================] - 115s 245ms/step - loss: 2.3222 - a
ccuracy: 0.1117 - val_loss: 2.3025 - val_accuracy: 0.1135
Test loss: 2.3025271892547607
Test accuracy: 0.11349999904632568
```

# Graphing Loss for Multi Layer Neural Network vs ConvNet with Noise

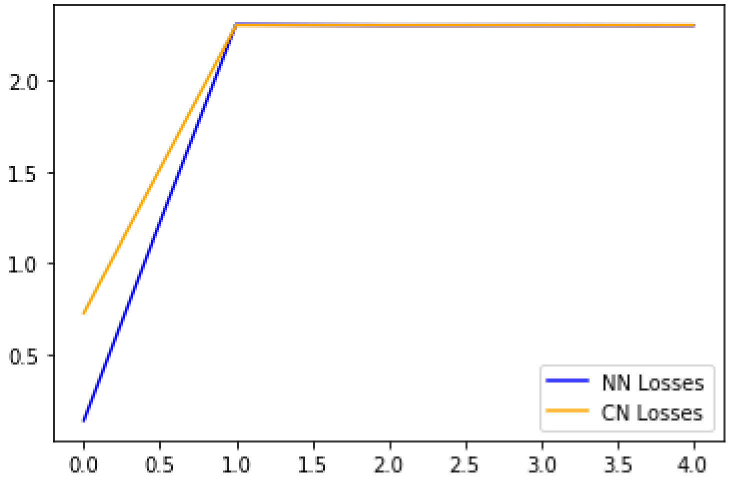
MLNN model has lower loss.

In [29]:

```
nn_losses = np.array([score_nn[0], score_nn10[0], score_nn15[0], score_nn20[0], s
cn_losses = np.array([score_cn[0], score_cn10[0], score_cn15[0], score_cn20[0], s
plt.plot(nn_losses, label='NN Losses', color='b')
plt.plot(cn_losses, label='CN Losses', color='orange')
plt.legend()
```

Out[29]:

```
<matplotlib.legend.Legend at 0x23283a866c8>
```



In [20]:

```
nn_losses
```

Out[20]:

```
array([0.13880014, 2.30621624, 2.30174112, 2.30239344, 2.30105662])
```

In [21]:

```
cn_losses
```

Out[21]:

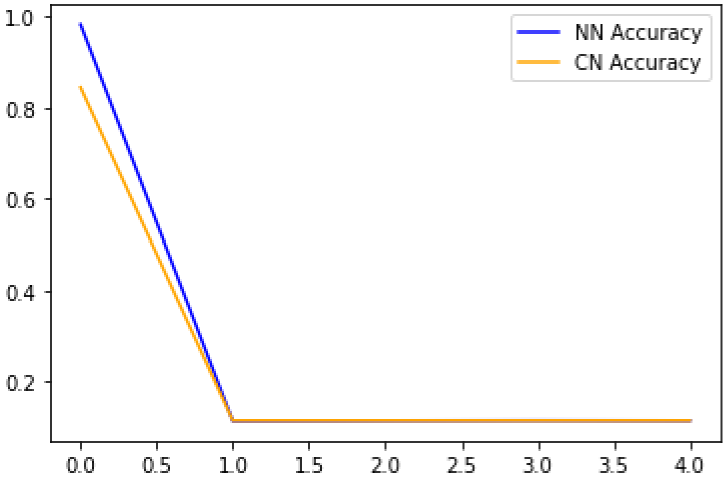
```
array([0.72762102, 2.30252361, 2.30252004, 2.30253148, 2.30252719])
```

# Graphing Accurancy for Multi Layer Neural Network vs ConvNet with Noise

MLNN model has higher accuracy loss.

```
In [30]: nn_accur = np.array([score_nn[1], score_nn10[1], score_nn15[1], score_nn20[1], sc
         cn_accur = np.array([score_cn[1], score_cn10[1], score_cn15[1], score_cn20[1], sc
         plt.plot(nn_accur, label='NN Accuracy', color='b')
         plt.plot(cn_accur, label='CN Accuracy', color='orange')
         plt.legend()
```

```
Out[30]: <matplotlib.legend.Legend at 0x23283ad2cc8>
```



```
In [31]: nn_accur
```

```
Out[31]: array([0.9835, 0.1128, 0.1135, 0.1138, 0.1135])
```

```
In [32]: cn_accur
```

```
Out[32]: array([0.84490001, 0.1135    , 0.1135    , 0.1135    , 0.1135    ])
```