Deep-Double-Descent

May 15, 2020

[1]:
```
#import libraries
import tensorflow as tf
import tensorflow.keras as K
from tensorflow.keras import losses, optimizers, metrics
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
from mlxtend.data import loadlocal_mnist
from tqdm import tqdm_notebook
```

[2]:
```
#download MNIST dataset
!curl -O http://yann.lecun.com/exdb/mnist/train-labels-idx1-ubyte.gz
!curl -O http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz
!curl -O http://yann.lecun.com/exdb/mnist/t10k-labels-idx1-ubyte.gz
!curl -O http://yann.lecun.com/exdb/mnist/t10k-images-idx3-ubyte.gz
```

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 28881  100 28881    0     0   159k      0 --:--:-- --:--:-- --:--:--  159k
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 9680k  100 9680k    0     0  22.9M      0 --:--:-- --:--:-- --:--:-- 22.9M
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  4542  100  4542    0     0  32212      0 --:--:-- --:--:-- --:--:-- 32212
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 1610k  100 1610k    0     0  5063k      0 --:--:-- --:--:-- --:--:-- 5047k
```

[0]:
```
#unzip files
!gunzip t*-ubyte.gz
```

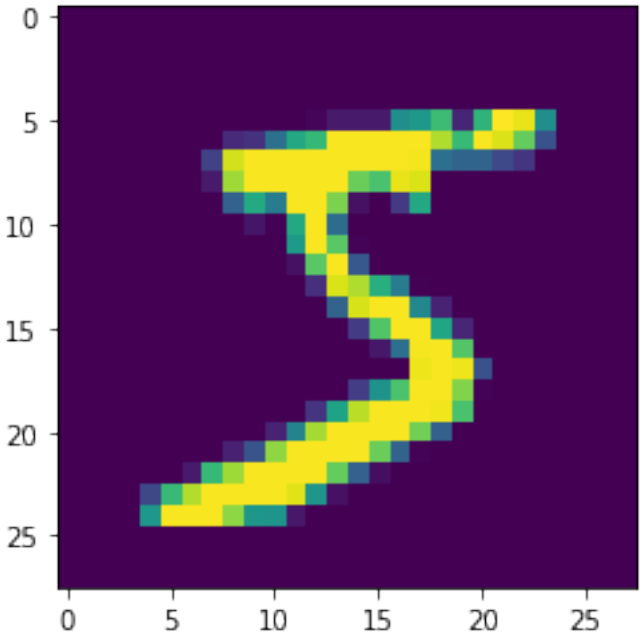
[0]:
```
#load dataset
X_train, y_train = loadlocal_mnist(
        images_path='train-images-idx3-ubyte',
        labels_path='train-labels-idx1-ubyte')
X_test, y_test = loadlocal_mnist(
```

```
        images_path='t10k-images-idx3-ubyte',
        labels_path='t10k-labels-idx1-ubyte')

#reshape to image shape
X_train = X_train.reshape(60000,28,28,1)
X_test = X_test.reshape(10000,28,28,1)
#sampled training dataset
X_train_6k = X_train.copy()[:6000]/255.0
y_train_6k = y_train.copy()[:6000]
#augmented training dataset
X_train_12k = X_train.copy()[:12000]/255.0
y_train_12k = y_train.copy()[:12000]
#sampled test dataset
X_test = X_test[:1000]/255.0
y_test = y_test[:1000]
```

[0]:
```
noise = 0.25 #25% noise

#pick random bernouli sample and assign fault label

bern = np.random.binomial(1,noise, len(y_train_6k))
s = sum(bern)
y_train_noise_6k = y_train_6k.copy()
y_train_noise_6k[bern==1] = np.random.randint(9,size=s)

bern = np.random.binomial(1,noise, len(y_train_12k))
s = sum(bern)
y_train_noise_12k = y_train_12k.copy()
y_train_noise_12k[bern==1] = np.random.randint(9,size=s)
```

[6]:
```
plt.imshow(X_train_6k[0].reshape(28,28))
```

[6]:
```
<matplotlib.image.AxesImage at 0x7fe5d936e198>
```



[7]:
```
y_train_6k[0]
```

[7]:
```
5
```

[0]:
```
#transform target output to one-hot-encoded representation
y_train_6k = tf.keras.utils.to_categorical(y_train_6k)
y_train_12k = tf.keras.utils.to_categorical(y_train_12k)
y_train_noise_6k = tf.keras.utils.to_categorical(y_train_noise_6k)
y_train_noise_12k = tf.keras.utils.to_categorical(y_train_noise_12k)

y_test = tf.keras.utils.to_categorical(y_test)
```

[9]:
```
y_train_6k[0]
```

[9]:
```
array([0., 0., 0., 0., 0., 1., 0., 0., 0., 0.], dtype=float32)
```

[0]:
```
#blocks definition
#input_size = 32 #ONLY FOR CIFAR DATASET

input_size = 28
def cnn_block(x_inp, filters, kernel_size=(3, 3),padding="same",␣
 ↪strides=1,pooling=2):
    '''function that returns a cnn block composed of Conv-BatchNorm-ReLU-MaxPool␣
 ↪layers'''
```

```
    x = K.layers.Conv2D(filters, kernel_size, padding=padding,␣
 ↪strides=strides)(x_inp)
    x = K.layers.BatchNormalization()(x)
    x = K.layers.Activation('relu')(x)
    x = K.layers.MaxPool2D(pool_size=(pooling,pooling))(x)
    return x


def resnet_block(x_inp, filters, kernel_size=(3, 3),padding="same",␣
 ↪strides=1,r=True):
    '''function that returns a cnn block composed of two␣
 ↪BatchNorm-ReLU-Convolution layers'''
    x = K.layers.BatchNormalization()(x_inp)
    x = K.layers.Activation('relu')(x)
    x = K.layers.Conv2D(filters, kernel_size, padding=padding,␣
 ↪strides=strides)(x)
    x = K.layers.BatchNormalization()(x)
    x = K.layers.Activation('relu')(x)
    x = K.layers.Conv2D(filters, kernel_size, padding=padding,␣
 ↪strides=strides)(x)
    if r:
        #interference Conv. layer in order to match the output dimensions
        x_inp = K.layers.Conv2D(filters,(1,1), padding=padding,␣
 ↪strides=strides)(x_inp)
        x_inp = K.layers.Conv2D(filters,(1,1), padding=padding,␣
 ↪strides=strides)(x_inp)
    x = K.layers.Add()([x,x_inp])
    return x


def resnet(k=64):
    '''ResNet network creation'''
    images = K.layers.Input((input_size,input_size, 1))
    x = resnet_block(images,k,strides=1)
    n_blocks = 4
    for i in range(1, n_blocks):
            x = resnet_block(x, k* (2**i),strides=2)
    x = K.layers.Flatten()(x)
    x = K.layers.Dense(10, activation='softmax')(x)
    net = K.models.Model(inputs=[images], outputs=[x])

    return(net)


def cnn(k=64):
    '''CNN network creation'''
```

```
    images = K.layers.Input((input_size,input_size, 1))
    x = cnn_block(images,k,pooling=1)
    n_blocks = 4
    for i in range(1, n_blocks):
        if i == 3:
            x = cnn_block(x, k* (2**i),pooling=4)
            #x = cnn_block(x, k * (2**i),pooling=8) only for CIFAR DATASET
        else:
            x = cnn_block(x, k * (2**i))
    x = K.layers.Flatten()(x)
    x = K.layers.Dense(10, activation='softmax')(x)
    net = K.models.Model(inputs=[images], outputs=[x])

    return(net)
```

[0]:
```
net = cnn()
net.summary()
```

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 28, 28, 1)]       0
_________________________________________________________________
conv2d (Conv2D)              (None, 28, 28, 64)        640
_________________________________________________________________
batch_normalization (BatchNo (None, 28, 28, 64)        256
_________________________________________________________________
activation (Activation)      (None, 28, 28, 64)        0
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 28, 28, 64)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 28, 28, 128)       73856
_________________________________________________________________
batch_normalization_1 (Batch (None, 28, 28, 128)       512
_________________________________________________________________
activation_1 (Activation)    (None, 28, 28, 128)       0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 14, 14, 128)       0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 14, 14, 256)       295168
_________________________________________________________________
batch_normalization_2 (Batch (None, 14, 14, 256)       1024
_________________________________________________________________
activation_2 (Activation)    (None, 14, 14, 256)       0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 7, 7, 256)         0
```

```
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 7, 7, 512)         1180160
_________________________________________________________________
batch_normalization_3 (Batch (None, 7, 7, 512)         2048
_________________________________________________________________
activation_3 (Activation)    (None, 7, 7, 512)         0
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 1, 1, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 512)               0
_________________________________________________________________
dense (Dense)                (None, 10)                5130
=================================================================
Total params: 1,558,794
Trainable params: 1,556,874
Non-trainable params: 1,920
_________________________________________________________________
```

[0]:
```
# from tensorflow.keras.utils import plot_model
# plot_model(net, to_file='model.png')
```

[0]:
```
net = resnet()
net.summary()
```

```
Model: "model_1"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_2 (InputLayer)            [(None, 28, 28, 1)]  0
__________________________________________________________________________________________________
batch_normalization_4 (BatchNor (None, 28, 28, 1)    4           input_2[0][0]
__________________________________________________________________________________________________
activation_4 (Activation)       (None, 28, 28, 1)    0
batch_normalization_4[0][0]
__________________________________________________________________________________________________
conv2d_4 (Conv2D)               (None, 28, 28, 64)   640
activation_4[0][0]
__________________________________________________________________________________________________
batch_normalization_5 (BatchNor (None, 28, 28, 64)   256         conv2d_4[0][0]
__________________________________________________________________________________________________
activation_5 (Activation)       (None, 28, 28, 64)   0
```

```
batch_normalization_5[0][0]
__________________________________________________________________________________________________
conv2d_6 (Conv2D)               (None, 28, 28, 64)   128         input_2[0][0]
__________________________________________________________________________________________________
conv2d_5 (Conv2D)               (None, 28, 28, 64)   36928
activation_5[0][0]
__________________________________________________________________________________________________
conv2d_7 (Conv2D)               (None, 28, 28, 64)   4160        conv2d_6[0][0]
__________________________________________________________________________________________________
add (Add)                       (None, 28, 28, 64)   0           conv2d_5[0][0]
                                                                 conv2d_7[0][0]
__________________________________________________________________________________________________
batch_normalization_6 (BatchNor (None, 28, 28, 64)   256         add[0][0]
__________________________________________________________________________________________________
activation_6 (Activation)       (None, 28, 28, 64)   0
batch_normalization_6[0][0]
__________________________________________________________________________________________________
conv2d_8 (Conv2D)               (None, 14, 14, 128)  73856
activation_6[0][0]
__________________________________________________________________________________________________
batch_normalization_7 (BatchNor (None, 14, 14, 128)  512         conv2d_8[0][0]
__________________________________________________________________________________________________
activation_7 (Activation)       (None, 14, 14, 128)  0
batch_normalization_7[0][0]
__________________________________________________________________________________________________
conv2d_10 (Conv2D)              (None, 14, 14, 128)  8320        add[0][0]
__________________________________________________________________________________________________
conv2d_9 (Conv2D)               (None, 7, 7, 128)    147584
activation_7[0][0]
__________________________________________________________________________________________________
conv2d_11 (Conv2D)              (None, 7, 7, 128)    16512       conv2d_10[0][0]
__________________________________________________________________________________________________
add_1 (Add)                     (None, 7, 7, 128)    0           conv2d_9[0][0]
                                                                 conv2d_11[0][0]
__________________________________________________________________________________________________
```

```
------------------
batch_normalization_8 (BatchNor (None, 7, 7, 128)    512         add_1[0][0]
--------------------------------------------------------------------------------
------------------
activation_8 (Activation)       (None, 7, 7, 128)    0
batch_normalization_8[0][0]
--------------------------------------------------------------------------------
------------------
conv2d_12 (Conv2D)              (None, 4, 4, 256)    295168
activation_8[0][0]
--------------------------------------------------------------------------------
------------------
batch_normalization_9 (BatchNor (None, 4, 4, 256)    1024        conv2d_12[0][0]
--------------------------------------------------------------------------------
------------------
activation_9 (Activation)       (None, 4, 4, 256)    0
batch_normalization_9[0][0]
--------------------------------------------------------------------------------
------------------
conv2d_14 (Conv2D)              (None, 4, 4, 256)    33024       add_1[0][0]
--------------------------------------------------------------------------------
------------------
conv2d_13 (Conv2D)              (None, 2, 2, 256)    590080
activation_9[0][0]
--------------------------------------------------------------------------------
------------------
conv2d_15 (Conv2D)              (None, 2, 2, 256)    65792       conv2d_14[0][0]
--------------------------------------------------------------------------------
------------------
add_2 (Add)                     (None, 2, 2, 256)    0           conv2d_13[0][0]
                                                                 conv2d_15[0][0]
--------------------------------------------------------------------------------
------------------
batch_normalization_10 (BatchNo (None, 2, 2, 256)    1024        add_2[0][0]
--------------------------------------------------------------------------------
------------------
activation_10 (Activation)      (None, 2, 2, 256)    0
batch_normalization_10[0][0]
--------------------------------------------------------------------------------
------------------
conv2d_16 (Conv2D)              (None, 1, 1, 512)    1180160
activation_10[0][0]
--------------------------------------------------------------------------------
------------------
batch_normalization_11 (BatchNo (None, 1, 1, 512)    2048        conv2d_16[0][0]
--------------------------------------------------------------------------------
------------------
activation_11 (Activation)      (None, 1, 1, 512)    0
```

```
batch_normalization_11[0][0]
__________________________________________________________________________________
__________________
conv2d_18 (Conv2D)              (None, 1, 1, 512)    131584      add_2[0][0]
__________________________________________________________________________________
__________________
conv2d_17 (Conv2D)              (None, 1, 1, 512)    2359808
activation_11[0][0]
__________________________________________________________________________________
__________________
conv2d_19 (Conv2D)              (None, 1, 1, 512)    262656      conv2d_18[0][0]
__________________________________________________________________________________
__________________
add_3 (Add)                     (None, 1, 1, 512)    0           conv2d_17[0][0]
                                                                 conv2d_19[0][0]
__________________________________________________________________________________
__________________
flatten_1 (Flatten)             (None, 512)          0           add_3[0][0]
__________________________________________________________________________________
__________________
dense_1 (Dense)                 (None, 10)           5130        flatten_1[0][0]
==================================================================================
==================
Total params: 5,217,166
Trainable params: 5,214,348
Non-trainable params: 2,818
__________________________________________________________________________________
__________________
```

[0]:
```
# from tensorflow.keras.utils import plot_model
# plot_model(net, to_file='model.png')
```

0.1 MODEL-WISE DOUBLE DESCENT

[0]:
```
train_loss_noiseless = {}
test_loss_noiseless = {}
train_loss_noise = {}
test_loss_noise = {}
for k in tqdm_notebook(range(1,21,1)):
    error_train_noiseless = []
    error_test_noiseless = []
    error_train_noise = []
    error_test_noise = []
    s = '_'
    for _ in range(3):
        #noiseless
        net = cnn(k)
```

```
        net.compile(loss=losses.CategoricalCrossentropy(), optimizer=optimizers.
↪Adam(0.001),metrics=metrics.categorical_accuracy)
        history = net.
↪fit(X_train_6k,y_train_6k,batch_size=128,epochs=100,validation_data=(X_test,y_test),verbose=0
        error_train_noiseless.append(history.history['loss'][-1])
        error_test_noiseless.append(history.history['val_loss'][-1])
        #noise
        net = cnn(k)
        net.compile(loss=losses.CategoricalCrossentropy(), optimizer=optimizers.
↪Adam(0.001),metrics=metrics.categorical_accuracy)
        history = net.
↪fit(X_train_6k,y_train_noise_6k,batch_size=128,epochs=100,validation_data=(X_test,y_test),ver
        error_train_noise.append(history.history['loss'][-1])
        error_test_noise.append(history.history['val_loss'][-1])
    train_loss_noiseless[k] = np.mean(error_train_noiseless)
    test_loss_noiseless[k] = np.mean(error_test_noiseless)
    train_loss_noise[k] = np.mean(error_train_noise)
    test_loss_noise[k] = np.mean(error_test_noise)

    if k > 1:
        if list(test_loss_noiseless.values())[-1] >= list(test_loss_noiseless.
↪values())[-2]:
            s = '+'
        else:
            s = '-'
    print('NOISELESS: k: {}, train_loss: {}, test_loss: {} // {}'.
↪format(k,round(train_loss_noiseless[k],4),round(test_loss_noiseless[k],4),s))
    if k > 1:
        if list(test_loss_noise.values())[-1] >= list(test_loss_noise.
↪values())[-2]:
            s = '+'
        else:
            s = '-'
    print('25% NOISE: k: {}, train_loss: {}, test_loss: {} // {}'.
↪format(k,round(train_loss_noise[k],4),round(test_loss_noise[k],4),s))
    print()
    print()
```

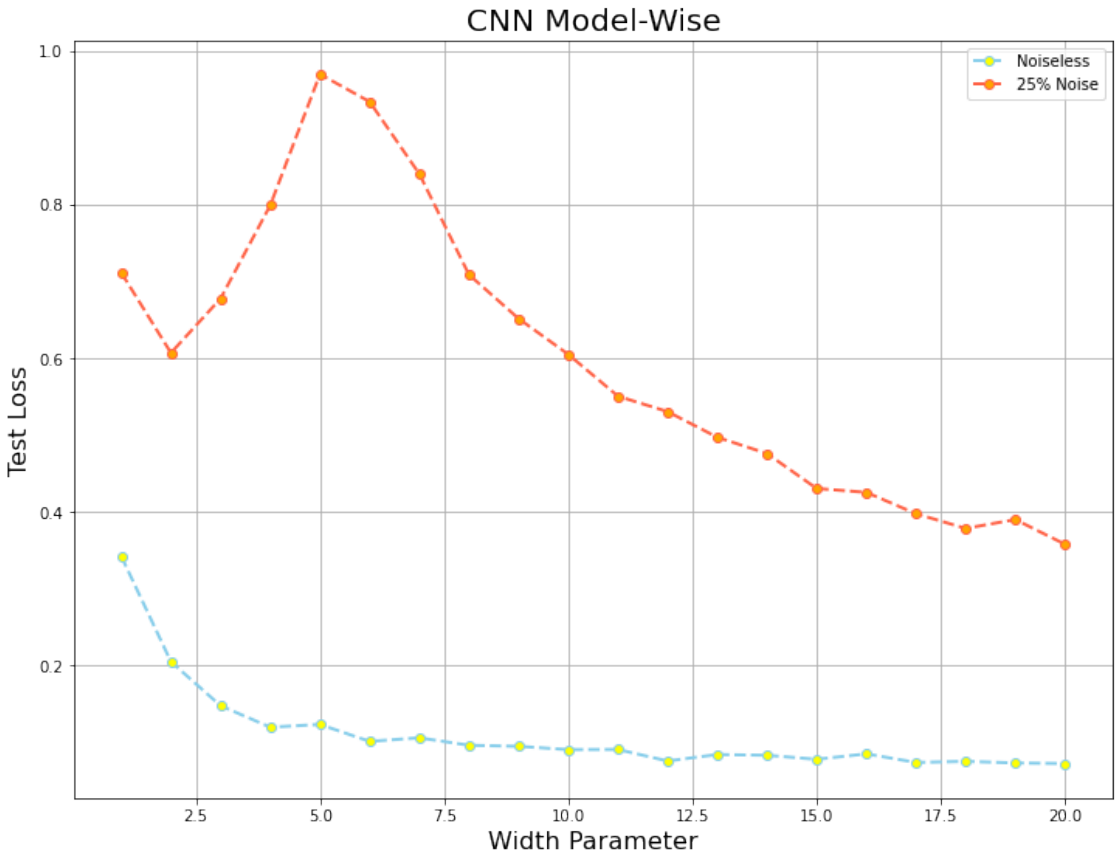
```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:5:
TqdmDeprecationWarning: This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
  """

HBox(children=(IntProgress(value=0, max=20), HTML(value='')))


NOISELESS: k: 1, train_loss: 0.2183, test_loss: 0.3422 // -
25% NOISE: k: 1, train_loss: 1.2479, test_loss: 0.7111 // -
```

```
NOISELESS: k: 2, train_loss: 0.021, test_loss: 0.205 // -
25% NOISE: k: 2, train_loss: 0.9697, test_loss: 0.6068 // -


NOISELESS: k: 3, train_loss: 0.0019, test_loss: 0.1477 // -
25% NOISE: k: 3, train_loss: 0.6987, test_loss: 0.6775 // +


NOISELESS: k: 4, train_loss: 0.0006, test_loss: 0.1199 // -
25% NOISE: k: 4, train_loss: 0.3972, test_loss: 0.7988 // +


NOISELESS: k: 5, train_loss: 0.0004, test_loss: 0.1235 // +
25% NOISE: k: 5, train_loss: 0.1234, test_loss: 0.9693 // +


NOISELESS: k: 6, train_loss: 0.0003, test_loss: 0.1014 // -
25% NOISE: k: 6, train_loss: 0.032, test_loss: 0.9335 // -


NOISELESS: k: 7, train_loss: 0.0002, test_loss: 0.1061 // +
25% NOISE: k: 7, train_loss: 0.0088, test_loss: 0.8397 // -


NOISELESS: k: 8, train_loss: 0.0001, test_loss: 0.0964 // -
25% NOISE: k: 8, train_loss: 0.0038, test_loss: 0.7085 // -


NOISELESS: k: 9, train_loss: 0.0001, test_loss: 0.095 // -
25% NOISE: k: 9, train_loss: 0.0024, test_loss: 0.6513 // -


NOISELESS: k: 10, train_loss: 0.0001, test_loss: 0.0907 // -
25% NOISE: k: 10, train_loss: 0.0015, test_loss: 0.6048 // -


NOISELESS: k: 11, train_loss: 0.0001, test_loss: 0.0909 // +
25% NOISE: k: 11, train_loss: 0.0011, test_loss: 0.5504 // -


NOISELESS: k: 12, train_loss: 0.0001, test_loss: 0.0759 // -
25% NOISE: k: 12, train_loss: 0.0008, test_loss: 0.5303 // -


NOISELESS: k: 13, train_loss: 0.0001, test_loss: 0.0843 // +
25% NOISE: k: 13, train_loss: 0.0007, test_loss: 0.497 // -
```

```
NOISELESS: k: 14, train_loss: 0.0001, test_loss: 0.0833 // -
25% NOISE: k: 14, train_loss: 0.0006, test_loss: 0.4759 // -


NOISELESS: k: 15, train_loss: 0.0, test_loss: 0.0781 // -
25% NOISE: k: 15, train_loss: 0.0005, test_loss: 0.4304 // -


NOISELESS: k: 16, train_loss: 0.0, test_loss: 0.0852 // +
25% NOISE: k: 16, train_loss: 0.0005, test_loss: 0.4254 // -


NOISELESS: k: 17, train_loss: 0.0, test_loss: 0.074 // -
25% NOISE: k: 17, train_loss: 0.0004, test_loss: 0.3976 // -


NOISELESS: k: 18, train_loss: 0.0, test_loss: 0.0755 // +
25% NOISE: k: 18, train_loss: 0.0003, test_loss: 0.3784 // -


NOISELESS: k: 19, train_loss: 0.0, test_loss: 0.0734 // -
25% NOISE: k: 19, train_loss: 0.0003, test_loss: 0.39 // +


NOISELESS: k: 20, train_loss: 0.0, test_loss: 0.0726 // -
25% NOISE: k: 20, train_loss: 0.0003, test_loss: 0.3578 // -
```

[0]:
```
# multiple line plot
plt.figure(figsize=(12,9))
plt.plot(list(test_loss_noiseless.keys()),list(test_loss_noiseless.
 ↪values()),'bx-',marker = '.',linestyle='--',
         markerfacecolor='yellow', color='skyblue',␣
 ↪linewidth=2,markersize=12,label='Noiseless')
plt.plot(list(test_loss_noise.keys()),list(test_loss_noise.
 ↪values()),'bx-',marker = '.',linestyle='--',
         markerfacecolor='orange', color='tomato',␣
 ↪linewidth=2,markersize=12,label='25% Noise')
plt.xlabel('Width Parameter',fontsize=16)
plt.ylabel('Test Loss',fontsize=16)
plt.title('CNN Model-Wise',fontsize=20)
plt.legend()
plt.grid()
```

```
plt.show()
```



0.2 SAMLE-WISE DOUBLE DESCENT

[0]:
```
train_loss_6k = {}
test_loss_6k = {}
train_loss_12k = {}
test_loss_12k = {}
for k in tqdm_notebook(range(1,21,1)):
    error_train_6k = []
    error_test_6k = []
    error_train_12k = []
    error_test_12k = []
    s = '-'
    for _ in range(3):
        #6k samples
        net = resnet(k)
        net.compile(loss=losses.CategoricalCrossentropy(), optimizer=optimizers.
↪Adam(0.001),metrics=metrics.categorical_accuracy)
```

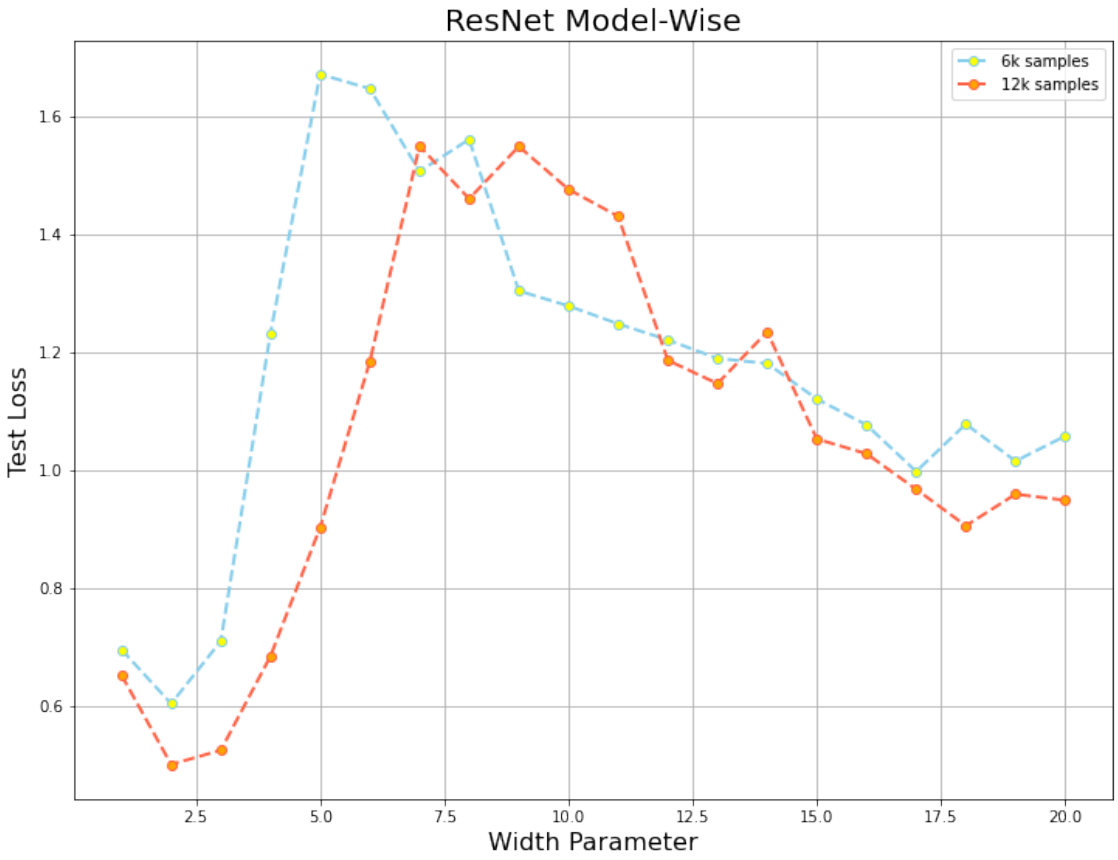
```
        history = net.
↪fit(X_train_6k,y_train_noise_6k,batch_size=128,epochs=60,validation_data=(X_test,y_test),verb
        error_train_6k.append(history.history['loss'][-1])
        error_test_6k.append(history.history['val_loss'][-1])
        #12k samples
        net = resnet(k)
        net.compile(loss=losses.CategoricalCrossentropy(), optimizer=optimizers.
↪Adam(0.001),metrics=metrics.categorical_accuracy)
        history = net.
↪fit(X_train_12k,y_train_noise_12k,batch_size=128,epochs=60,validation_data=(X_test,y_test),ve
        error_train_12k.append(history.history['loss'][-1])
        error_test_12k.append(history.history['val_loss'][-1])
    train_loss_6k[k] = np.mean(error_train_6k)
    test_loss_6k[k] = np.mean(error_test_6k)
    train_loss_12k[k] = np.mean(error_train_12k)
    test_loss_12k[k] = np.mean(error_test_12k)

    if k > 1:
        if list(test_loss_6k.values())[-1] >= list(test_loss_6k.values())[-2]:
            s = '+'
        else:
            s = '-'
    print('6k SAMPLES: k: {}, train_loss: {}, test_loss: {} // {}'.
↪format(k,round(train_loss_6k[k],4),round(test_loss_6k[k],4),s))
    if k > 1:
        if list(test_loss_12k.values())[-1] >= list(test_loss_12k.values())[-2]:
            s = '+'
        else:
            s = '-'
    print('12k SAMPLES: k: {}, train_loss: {}, test_loss: {} // {}'.
↪format(k,round(train_loss_12k[k],4),round(test_loss_12k[k],4),s))
    print()
    print()
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:5:
TqdmDeprecationWarning: This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
  """

HBox(children=(IntProgress(value=0, max=20), HTML(value='')))


6k SAMPLES: k: 1, train_loss: 1.1909, test_loss: 0.6954 // -
12k SAMPLES: k: 1, train_loss: 1.2336, test_loss: 0.6522 // -


6k SAMPLES: k: 2, train_loss: 0.9735, test_loss: 0.6037 // -
12k SAMPLES: k: 2, train_loss: 1.0519, test_loss: 0.5006 // -
```

```
6k SAMPLES: k: 3, train_loss: 0.735, test_loss: 0.7096 // +
12k SAMPLES: k: 3, train_loss: 0.9162, test_loss: 0.5247 // +


6k SAMPLES: k: 4, train_loss: 0.3214, test_loss: 1.2316 // +
12k SAMPLES: k: 4, train_loss: 0.7196, test_loss: 0.6841 // +


6k SAMPLES: k: 5, train_loss: 0.15, test_loss: 1.6702 // +
12k SAMPLES: k: 5, train_loss: 0.515, test_loss: 0.9023 // +


6k SAMPLES: k: 6, train_loss: 0.0845, test_loss: 1.6459 // -
12k SAMPLES: k: 6, train_loss: 0.3006, test_loss: 1.1831 // +


6k SAMPLES: k: 7, train_loss: 0.0083, test_loss: 1.5057 // -
12k SAMPLES: k: 7, train_loss: 0.1703, test_loss: 1.5486 // +


6k SAMPLES: k: 8, train_loss: 0.0015, test_loss: 1.5599 // +
12k SAMPLES: k: 8, train_loss: 0.1381, test_loss: 1.4593 // -


6k SAMPLES: k: 9, train_loss: 0.0005, test_loss: 1.3032 // -
12k SAMPLES: k: 9, train_loss: 0.121, test_loss: 1.5479 // +


6k SAMPLES: k: 10, train_loss: 0.0003, test_loss: 1.2777 // -
12k SAMPLES: k: 10, train_loss: 0.0771, test_loss: 1.4752 // -


6k SAMPLES: k: 11, train_loss: 0.0003, test_loss: 1.2473 // -
12k SAMPLES: k: 11, train_loss: 0.0793, test_loss: 1.4295 // -


6k SAMPLES: k: 12, train_loss: 0.0002, test_loss: 1.2199 // -
12k SAMPLES: k: 12, train_loss: 0.0005, test_loss: 1.1855 // -


6k SAMPLES: k: 13, train_loss: 0.0002, test_loss: 1.1888 // -
12k SAMPLES: k: 13, train_loss: 0.0003, test_loss: 1.1461 // -


6k SAMPLES: k: 14, train_loss: 0.0001, test_loss: 1.1802 // -
12k SAMPLES: k: 14, train_loss: 0.0002, test_loss: 1.2338 // +
```

```
6k SAMPLES: k: 15, train_loss: 0.0001, test_loss: 1.12 // -
12k SAMPLES: k: 15, train_loss: 0.0003, test_loss: 1.0522 // -


6k SAMPLES: k: 16, train_loss: 0.0001, test_loss: 1.076 // -
12k SAMPLES: k: 16, train_loss: 0.0002, test_loss: 1.0272 // -


6k SAMPLES: k: 17, train_loss: 0.0001, test_loss: 0.9971 // -
12k SAMPLES: k: 17, train_loss: 0.0002, test_loss: 0.9669 // -


6k SAMPLES: k: 18, train_loss: 0.0001, test_loss: 1.0775 // +
12k SAMPLES: k: 18, train_loss: 0.0002, test_loss: 0.9052 // -


6k SAMPLES: k: 19, train_loss: 0.0001, test_loss: 1.0147 // -
12k SAMPLES: k: 19, train_loss: 0.0001, test_loss: 0.9587 // +


6k SAMPLES: k: 20, train_loss: 0.0001, test_loss: 1.0572 // +
12k SAMPLES: k: 20, train_loss: 0.0001, test_loss: 0.9483 // -
```

[0]:
```
# multiple line plot
plt.figure(figsize=(12,9))
plt.plot(list(test_loss_6k.keys()),list(test_loss_6k.values()),'bx-',marker = '.
 ↪',linestyle='--',
         markerfacecolor='yellow', color='skyblue',␣
 ↪linewidth=2,markersize=12,label='6k samples')
plt.plot(list(test_loss_12k.keys()),list(test_loss_12k.values()),'bx-',marker =␣
 ↪'.',linestyle='--',
         markerfacecolor='orange', color='tomato',␣
 ↪linewidth=2,markersize=12,label='12k samples')
plt.xlabel('Width Parameter',fontsize=16)
plt.ylabel('Test Loss',fontsize=16)
plt.title('ResNet Model-Wise',fontsize=20)
plt.legend()
plt.grid()
plt.show()
```



0.3 EPOCH-WISE DOUBLE DESCENT

[11]:
```
train_loss = {}
test_loss = {}
for k in [3,12,48]:
    print()
    print('Model width parameter k: {}'.format(k))
    train_loss[k] = {}
    test_loss[k] = {}
    for e in tqdm_notebook(range(5,101,1)):
        error_train = []
        error_test = []
        s = '-'
        for _ in range(3):
            net = resnet(k)
            net.compile(loss=losses.CategoricalCrossentropy(),␣
↪optimizer=optimizers.Adam(0.001),metrics=metrics.categorical_accuracy)
            history = net.
↪fit(X_train_6k,y_train_noise_6k,batch_size=128,epochs=e,validation_data=(X_test,y_test),verbo
```

```
            error_train.append(history.history['loss'][-1])
            error_test.append(history.history['val_loss'][-1])
        train_loss[k][e] = np.mean(error_train)
        test_loss[k][e] = np.mean(error_test)
        if e > 5:
            if list(test_loss[k].values())[-1] >= list(test_loss[k].
↪values())[-2]:
                s = '+'
            else:
                s = '-'
        print('Epoch: {}, train_loss: {}, test_loss: {} // {}'.
↪format(e,round(train_loss[k][e],4),round(test_loss[k][e],4),s))
```

```
Model width parameter k: 3
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:8:
TqdmDeprecationWarning: This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
```

```
HBox(children=(IntProgress(value=0, max=96), HTML(value='')))
```

```
Epoch: 5, train_loss: 1.3091, test_loss: 1.4248 // -
Epoch: 6, train_loss: 1.2631, test_loss: 1.0099 // -
Epoch: 7, train_loss: 1.1993, test_loss: 0.9096 // -
Epoch: 8, train_loss: 1.1872, test_loss: 0.8674 // -
Epoch: 9, train_loss: 1.1722, test_loss: 0.7831 // -
Epoch: 10, train_loss: 1.1636, test_loss: 0.7019 // -
Epoch: 11, train_loss: 1.1371, test_loss: 0.6908 // -
Epoch: 12, train_loss: 1.1197, test_loss: 0.658 // -
Epoch: 13, train_loss: 1.1125, test_loss: 0.6547 // -
Epoch: 14, train_loss: 1.0921, test_loss: 0.5883 // -
Epoch: 15, train_loss: 1.0726, test_loss: 0.6029 // +
Epoch: 16, train_loss: 1.0602, test_loss: 0.6012 // -
Epoch: 17, train_loss: 1.0462, test_loss: 0.615 // +
Epoch: 18, train_loss: 1.0549, test_loss: 0.6247 // +
Epoch: 19, train_loss: 1.0349, test_loss: 0.5998 // -
Epoch: 20, train_loss: 1.0301, test_loss: 0.5991 // -
Epoch: 21, train_loss: 1.0031, test_loss: 0.6342 // +
Epoch: 22, train_loss: 1.0032, test_loss: 0.6049 // -
Epoch: 23, train_loss: 0.9907, test_loss: 0.5863 // -
Epoch: 24, train_loss: 0.9868, test_loss: 0.5624 // -
Epoch: 25, train_loss: 0.9615, test_loss: 0.6007 // +
Epoch: 26, train_loss: 0.9688, test_loss: 0.5475 // -
Epoch: 27, train_loss: 0.9645, test_loss: 0.5562 // +
Epoch: 28, train_loss: 0.9467, test_loss: 0.6278 // +
Epoch: 29, train_loss: 0.941, test_loss: 0.5964 // -
```

```
Epoch: 30, train_loss: 0.9246, test_loss: 0.5684 // -
Epoch: 31, train_loss: 0.9122, test_loss: 0.6142 // +
Epoch: 32, train_loss: 0.9094, test_loss: 0.6472 // +
Epoch: 33, train_loss: 0.9151, test_loss: 0.6047 // -
Epoch: 34, train_loss: 0.9171, test_loss: 0.6413 // +
Epoch: 35, train_loss: 0.9087, test_loss: 0.6091 // -
Epoch: 36, train_loss: 0.8918, test_loss: 0.6257 // +
Epoch: 37, train_loss: 0.8912, test_loss: 0.6309 // +
Epoch: 38, train_loss: 0.8645, test_loss: 0.6468 // +
Epoch: 39, train_loss: 0.844, test_loss: 0.6411 // -
Epoch: 40, train_loss: 0.8508, test_loss: 0.7007 // +
Epoch: 41, train_loss: 0.8682, test_loss: 0.6235 // -
Epoch: 42, train_loss: 0.8529, test_loss: 0.6148 // -
Epoch: 43, train_loss: 0.8492, test_loss: 0.6302 // +
Epoch: 44, train_loss: 0.8314, test_loss: 0.6706 // +
Epoch: 45, train_loss: 0.8311, test_loss: 0.6495 // -
Epoch: 46, train_loss: 0.8168, test_loss: 0.717 // +
Epoch: 47, train_loss: 0.8002, test_loss: 0.8605 // +
Epoch: 48, train_loss: 0.8169, test_loss: 0.6667 // -
Epoch: 49, train_loss: 0.8096, test_loss: 0.69 // +
Epoch: 50, train_loss: 0.7947, test_loss: 0.727 // +
Epoch: 51, train_loss: 0.7797, test_loss: 0.6645 // -
Epoch: 52, train_loss: 0.7792, test_loss: 0.7192 // +
Epoch: 53, train_loss: 0.7893, test_loss: 0.6834 // -
Epoch: 54, train_loss: 0.7665, test_loss: 0.7887 // +
Epoch: 55, train_loss: 0.7835, test_loss: 0.7024 // -
Epoch: 56, train_loss: 0.7669, test_loss: 0.6776 // -
Epoch: 57, train_loss: 0.7574, test_loss: 0.7269 // +
Epoch: 58, train_loss: 0.7694, test_loss: 0.7454 // +
Epoch: 59, train_loss: 0.761, test_loss: 0.7278 // -
Epoch: 60, train_loss: 0.7569, test_loss: 0.7511 // +
Epoch: 61, train_loss: 0.752, test_loss: 0.7488 // -
Epoch: 62, train_loss: 0.742, test_loss: 0.799 // +
Epoch: 63, train_loss: 0.7234, test_loss: 0.7623 // -
Epoch: 64, train_loss: 0.7311, test_loss: 0.7611 // -
Epoch: 65, train_loss: 0.7251, test_loss: 0.7555 // -
Epoch: 66, train_loss: 0.7202, test_loss: 0.7691 // +
Epoch: 67, train_loss: 0.7299, test_loss: 0.7723 // +
Epoch: 68, train_loss: 0.7052, test_loss: 0.793 // +
Epoch: 69, train_loss: 0.7225, test_loss: 0.7886 // -
Epoch: 70, train_loss: 0.7369, test_loss: 0.7487 // -
Epoch: 71, train_loss: 0.6956, test_loss: 0.7777 // +
Epoch: 72, train_loss: 0.6733, test_loss: 0.8324 // +
Epoch: 73, train_loss: 0.6848, test_loss: 0.8219 // -
Epoch: 74, train_loss: 0.6975, test_loss: 0.8524 // +
Epoch: 75, train_loss: 0.6866, test_loss: 0.8455 // -
Epoch: 76, train_loss: 0.6954, test_loss: 0.7813 // -
Epoch: 77, train_loss: 0.6827, test_loss: 0.8366 // +
```

```
Epoch: 78, train_loss: 0.6706, test_loss: 0.9704 // +
Epoch: 79, train_loss: 0.6579, test_loss: 0.8836 // -
Epoch: 80, train_loss: 0.6671, test_loss: 0.833 // -
Epoch: 81, train_loss: 0.6894, test_loss: 0.8365 // +
Epoch: 82, train_loss: 0.6803, test_loss: 0.8189 // -
Epoch: 83, train_loss: 0.6632, test_loss: 0.8239 // +
Epoch: 84, train_loss: 0.6671, test_loss: 0.8103 // -
Epoch: 85, train_loss: 0.6702, test_loss: 0.8582 // +
Epoch: 86, train_loss: 0.6558, test_loss: 0.9216 // +
Epoch: 87, train_loss: 0.6516, test_loss: 0.924 // +
Epoch: 88, train_loss: 0.6616, test_loss: 0.8997 // -
Epoch: 89, train_loss: 0.6764, test_loss: 0.8933 // -
Epoch: 90, train_loss: 0.6377, test_loss: 0.9264 // +
Epoch: 91, train_loss: 0.6504, test_loss: 0.837 // -
Epoch: 92, train_loss: 0.64, test_loss: 0.904 // +
Epoch: 93, train_loss: 0.6252, test_loss: 0.8566 // -
Epoch: 94, train_loss: 0.6332, test_loss: 0.9776 // +
Epoch: 95, train_loss: 0.6165, test_loss: 0.9229 // -
Epoch: 96, train_loss: 0.6156, test_loss: 0.9459 // +
Epoch: 97, train_loss: 0.6264, test_loss: 0.9004 // -
Epoch: 98, train_loss: 0.622, test_loss: 0.8935 // -
Epoch: 99, train_loss: 0.6321, test_loss: 0.8686 // -
Epoch: 100, train_loss: 0.6267, test_loss: 0.943 // +


Model width parameter k: 12
HBox(children=(IntProgress(value=0, max=96), HTML(value='')))


Epoch: 5, train_loss: 0.8719, test_loss: 1.6419 // -
Epoch: 6, train_loss: 0.7773, test_loss: 1.3107 // -
Epoch: 7, train_loss: 0.6562, test_loss: 0.9981 // -
Epoch: 8, train_loss: 0.5298, test_loss: 1.0522 // +
Epoch: 9, train_loss: 0.4314, test_loss: 0.9563 // -
Epoch: 10, train_loss: 0.3586, test_loss: 0.9038 // -
Epoch: 11, train_loss: 0.2351, test_loss: 1.021 // +
Epoch: 12, train_loss: 0.208, test_loss: 1.0364 // +
Epoch: 13, train_loss: 0.1734, test_loss: 1.0075 // -
Epoch: 14, train_loss: 0.1332, test_loss: 1.085 // +
Epoch: 15, train_loss: 0.1252, test_loss: 1.1145 // +
Epoch: 16, train_loss: 0.0872, test_loss: 1.1737 // +
Epoch: 17, train_loss: 0.0807, test_loss: 1.1601 // -
Epoch: 18, train_loss: 0.0813, test_loss: 1.2609 // +
Epoch: 19, train_loss: 0.1166, test_loss: 1.1926 // -
Epoch: 20, train_loss: 0.102, test_loss: 1.412 // +
Epoch: 21, train_loss: 0.1027, test_loss: 1.4116 // -
Epoch: 22, train_loss: 0.0999, test_loss: 1.3284 // -
Epoch: 23, train_loss: 0.0972, test_loss: 1.4839 // +
```

```
Epoch: 24, train_loss: 0.0574, test_loss: 1.4573 // -
Epoch: 25, train_loss: 0.0839, test_loss: 1.3677 // -
Epoch: 26, train_loss: 0.0216, test_loss: 1.229 // -
Epoch: 27, train_loss: 0.0147, test_loss: 1.1802 // -
Epoch: 28, train_loss: 0.006, test_loss: 1.2167 // +
Epoch: 29, train_loss: 0.0546, test_loss: 1.377 // +
Epoch: 30, train_loss: 0.0088, test_loss: 1.2249 // -
Epoch: 31, train_loss: 0.0053, test_loss: 1.2652 // +
Epoch: 32, train_loss: 0.0014, test_loss: 1.2059 // -
Epoch: 33, train_loss: 0.0008, test_loss: 1.2545 // +
Epoch: 34, train_loss: 0.001, test_loss: 1.126 // -
Epoch: 35, train_loss: 0.0035, test_loss: 1.1621 // +
Epoch: 36, train_loss: 0.0007, test_loss: 1.1565 // -
Epoch: 37, train_loss: 0.0007, test_loss: 1.1725 // +
Epoch: 38, train_loss: 0.0006, test_loss: 1.2385 // +
Epoch: 39, train_loss: 0.0068, test_loss: 1.331 // +
Epoch: 40, train_loss: 0.0005, test_loss: 1.1765 // -
Epoch: 41, train_loss: 0.0005, test_loss: 1.3192 // +
Epoch: 42, train_loss: 0.0006, test_loss: 1.1421 // -
Epoch: 43, train_loss: 0.0004, test_loss: 1.2119 // +
Epoch: 44, train_loss: 0.0004, test_loss: 1.2793 // +
Epoch: 45, train_loss: 0.0004, test_loss: 1.2668 // -
Epoch: 46, train_loss: 0.0005, test_loss: 1.2051 // -
Epoch: 47, train_loss: 0.0003, test_loss: 1.2771 // +
Epoch: 48, train_loss: 0.0003, test_loss: 1.2163 // -
Epoch: 49, train_loss: 0.0003, test_loss: 1.167 // -
Epoch: 50, train_loss: 0.0003, test_loss: 1.2633 // +
Epoch: 51, train_loss: 0.0003, test_loss: 1.2678 // +
Epoch: 52, train_loss: 0.0003, test_loss: 1.2822 // +
Epoch: 53, train_loss: 0.0003, test_loss: 1.1925 // -
Epoch: 54, train_loss: 0.0003, test_loss: 1.2654 // +
Epoch: 55, train_loss: 0.0003, test_loss: 1.2571 // -
Epoch: 56, train_loss: 0.0002, test_loss: 1.2656 // +
Epoch: 57, train_loss: 0.0002, test_loss: 1.3012 // +
Epoch: 58, train_loss: 0.0002, test_loss: 1.2962 // -
Epoch: 59, train_loss: 0.0002, test_loss: 1.1728 // -
Epoch: 60, train_loss: 0.0002, test_loss: 1.4331 // +
Epoch: 61, train_loss: 0.0002, test_loss: 1.2895 // -
Epoch: 62, train_loss: 0.0002, test_loss: 1.2785 // -
Epoch: 63, train_loss: 0.0002, test_loss: 1.3371 // +
Epoch: 64, train_loss: 0.0002, test_loss: 1.3094 // -
Epoch: 65, train_loss: 0.0002, test_loss: 1.2624 // -
Epoch: 66, train_loss: 0.0002, test_loss: 1.3801 // +
Epoch: 67, train_loss: 0.0001, test_loss: 1.3802 // +
Epoch: 68, train_loss: 0.0001, test_loss: 1.3535 // -
Epoch: 69, train_loss: 0.0001, test_loss: 1.3567 // +
Epoch: 70, train_loss: 0.0001, test_loss: 1.3748 // +
Epoch: 71, train_loss: 0.0001, test_loss: 1.2885 // -
```

```
Epoch: 72, train_loss: 0.0001, test_loss: 1.3197 // +
Epoch: 73, train_loss: 0.0001, test_loss: 1.2638 // -
Epoch: 74, train_loss: 0.0001, test_loss: 1.3005 // +
Epoch: 75, train_loss: 0.0003, test_loss: 1.2976 // -
Epoch: 76, train_loss: 0.0001, test_loss: 1.3152 // +
Epoch: 77, train_loss: 0.0001, test_loss: 1.3591 // +
Epoch: 78, train_loss: 0.0001, test_loss: 1.2672 // -
Epoch: 79, train_loss: 0.0001, test_loss: 1.4628 // +
Epoch: 80, train_loss: 0.0001, test_loss: 1.191 // -
Epoch: 81, train_loss: 0.0001, test_loss: 1.2741 // +
Epoch: 82, train_loss: 0.0001, test_loss: 1.4161 // +
Epoch: 83, train_loss: 0.0001, test_loss: 1.3183 // -
Epoch: 84, train_loss: 0.0001, test_loss: 1.4026 // +
Epoch: 85, train_loss: 0.0001, test_loss: 1.3238 // -
Epoch: 86, train_loss: 0.0001, test_loss: 1.3966 // +
Epoch: 87, train_loss: 0.0001, test_loss: 1.4126 // +
Epoch: 88, train_loss: 0.0001, test_loss: 1.2959 // -
Epoch: 89, train_loss: 0.0001, test_loss: 1.3464 // +
Epoch: 90, train_loss: 0.0001, test_loss: 1.1803 // -
Epoch: 91, train_loss: 0.0001, test_loss: 1.3415 // +
Epoch: 92, train_loss: 0.0001, test_loss: 1.3581 // +
Epoch: 93, train_loss: 0.0001, test_loss: 1.3516 // -
Epoch: 94, train_loss: 0.0001, test_loss: 1.4128 // +
Epoch: 95, train_loss: 0.0001, test_loss: 1.3571 // -
Epoch: 96, train_loss: 0.0001, test_loss: 1.3293 // -
Epoch: 97, train_loss: 0.0001, test_loss: 1.4227 // +
Epoch: 98, train_loss: 0.0001, test_loss: 1.2989 // -
Epoch: 99, train_loss: 0.0001, test_loss: 1.4346 // +
Epoch: 100, train_loss: 0.0001, test_loss: 1.3577 // -


Model width parameter k: 48
HBox(children=(IntProgress(value=0, max=96), HTML(value='')))


Epoch: 5, train_loss: 0.8032, test_loss: 2.2161 // -
Epoch: 6, train_loss: 0.6598, test_loss: 1.6302 // -
Epoch: 7, train_loss: 0.4665, test_loss: 1.1643 // -
Epoch: 8, train_loss: 0.3257, test_loss: 1.0377 // -
Epoch: 9, train_loss: 0.2327, test_loss: 1.3734 // +
Epoch: 10, train_loss: 0.1929, test_loss: 0.9158 // -
Epoch: 11, train_loss: 0.1858, test_loss: 1.0798 // +
Epoch: 12, train_loss: 0.16, test_loss: 1.2029 // +
Epoch: 13, train_loss: 0.1216, test_loss: 1.2335 // +
Epoch: 14, train_loss: 0.1274, test_loss: 1.2078 // -
Epoch: 15, train_loss: 0.1146, test_loss: 1.2356 // +
Epoch: 16, train_loss: 0.1071, test_loss: 1.1094 // -
Epoch: 17, train_loss: 0.0928, test_loss: 1.3955 // +
```

```
Epoch: 18, train_loss: 0.0717, test_loss: 1.3584 // -
Epoch: 19, train_loss: 0.1141, test_loss: 1.4896 // +
Epoch: 20, train_loss: 0.0836, test_loss: 1.3348 // -
Epoch: 21, train_loss: 0.0657, test_loss: 1.4229 // +
Epoch: 22, train_loss: 0.0754, test_loss: 1.3539 // -
Epoch: 23, train_loss: 0.0716, test_loss: 1.4796 // +
Epoch: 24, train_loss: 0.034, test_loss: 1.5171 // +
Epoch: 25, train_loss: 0.0433, test_loss: 1.2576 // -
Epoch: 26, train_loss: 0.0514, test_loss: 1.3253 // +
Epoch: 27, train_loss: 0.0673, test_loss: 1.5725 // +
Epoch: 28, train_loss: 0.0626, test_loss: 1.2034 // -
Epoch: 29, train_loss: 0.0434, test_loss: 1.219 // +
Epoch: 30, train_loss: 0.034, test_loss: 1.3293 // +
Epoch: 31, train_loss: 0.0318, test_loss: 1.2051 // -
Epoch: 32, train_loss: 0.0173, test_loss: 1.1377 // -
Epoch: 33, train_loss: 0.0247, test_loss: 1.3095 // +
Epoch: 34, train_loss: 0.0796, test_loss: 1.5184 // +
Epoch: 35, train_loss: 0.0298, test_loss: 1.4265 // -
Epoch: 36, train_loss: 0.0299, test_loss: 1.2304 // -
Epoch: 37, train_loss: 0.0153, test_loss: 1.2747 // +
Epoch: 38, train_loss: 0.0187, test_loss: 1.1933 // -
Epoch: 39, train_loss: 0.0274, test_loss: 1.2443 // +
Epoch: 40, train_loss: 0.0226, test_loss: 1.2128 // -
Epoch: 41, train_loss: 0.0204, test_loss: 1.2979 // +
Epoch: 42, train_loss: 0.0266, test_loss: 1.3481 // +
Epoch: 43, train_loss: 0.0001, test_loss: 1.085 // -
Epoch: 44, train_loss: 0.0003, test_loss: 1.1258 // +
Epoch: 45, train_loss: 0.008, test_loss: 1.1202 // -
Epoch: 46, train_loss: 0.007, test_loss: 1.2711 // +
Epoch: 47, train_loss: 0.0146, test_loss: 1.2122 // -
Epoch: 48, train_loss: 0.0002, test_loss: 1.143 // -
Epoch: 49, train_loss: 0.0001, test_loss: 1.0219 // -
Epoch: 50, train_loss: 0.0069, test_loss: 1.2434 // +
Epoch: 51, train_loss: 0.0161, test_loss: 1.4628 // +
Epoch: 52, train_loss: 0.0, test_loss: 1.1178 // -
Epoch: 53, train_loss: 0.001, test_loss: 1.117 // -
Epoch: 54, train_loss: 0.0, test_loss: 1.1647 // +
Epoch: 55, train_loss: 0.0026, test_loss: 1.2123 // +
Epoch: 56, train_loss: 0.0067, test_loss: 1.2664 // +
Epoch: 57, train_loss: 0.0014, test_loss: 1.0778 // -
Epoch: 58, train_loss: 0.0, test_loss: 1.0591 // -
Epoch: 59, train_loss: 0.0, test_loss: 1.1752 // +
Epoch: 60, train_loss: 0.0001, test_loss: 1.1158 // -
Epoch: 61, train_loss: 0.0, test_loss: 1.0322 // -
Epoch: 62, train_loss: 0.0, test_loss: 1.0439 // +
Epoch: 63, train_loss: 0.0, test_loss: 1.1248 // +
Epoch: 64, train_loss: 0.0, test_loss: 1.2579 // +
Epoch: 65, train_loss: 0.0, test_loss: 1.1824 // -
```

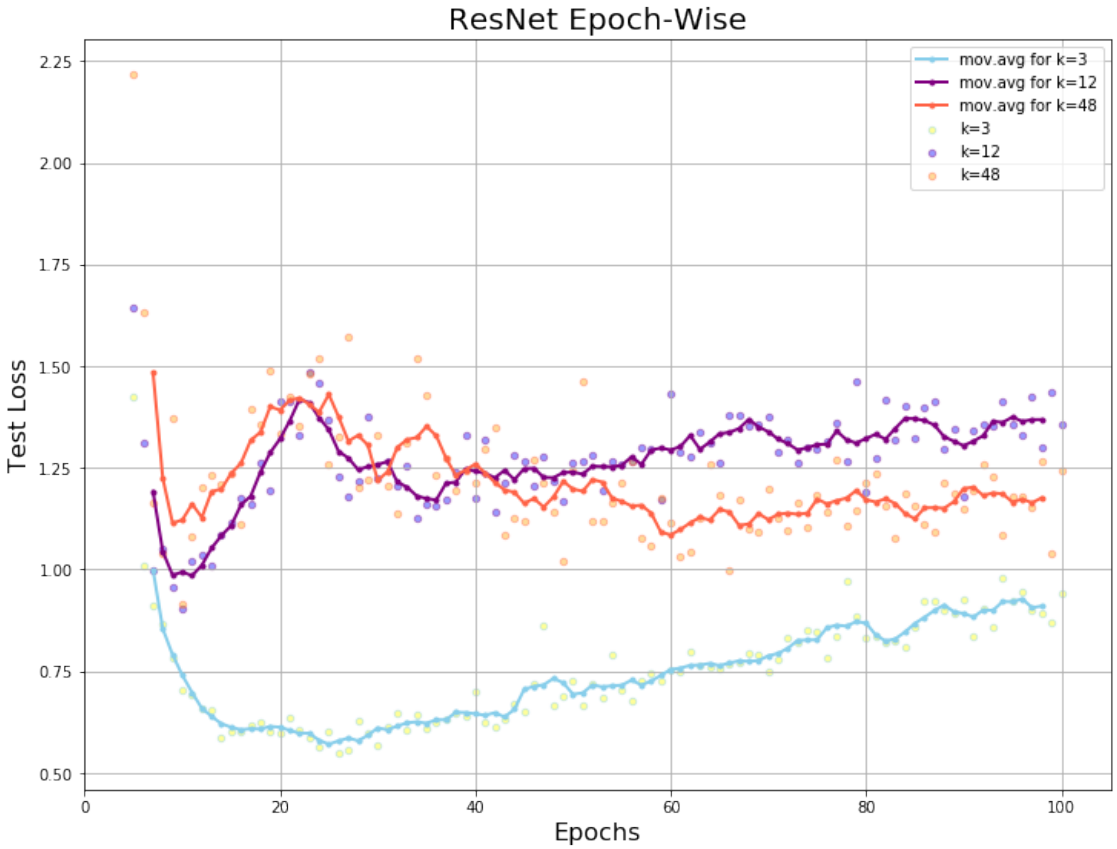
```
Epoch: 66, train_loss: 0.0, test_loss: 0.9997 // -
Epoch: 67, train_loss: 0.0, test_loss: 1.1719 // +
Epoch: 68, train_loss: 0.0, test_loss: 1.0981 // -
Epoch: 69, train_loss: 0.0, test_loss: 1.0912 // -
Epoch: 70, train_loss: 0.0, test_loss: 1.1987 // +
Epoch: 71, train_loss: 0.0, test_loss: 1.1276 // -
Epoch: 72, train_loss: 0.0, test_loss: 1.0979 // -
Epoch: 73, train_loss: 0.0, test_loss: 1.1644 // +
Epoch: 74, train_loss: 0.0, test_loss: 1.1054 // -
Epoch: 75, train_loss: 0.0, test_loss: 1.1842 // +
Epoch: 76, train_loss: 0.0, test_loss: 1.142 // -
Epoch: 77, train_loss: 0.0, test_loss: 1.2697 // +
Epoch: 78, train_loss: 0.0, test_loss: 1.1067 // -
Epoch: 79, train_loss: 0.0, test_loss: 1.145 // +
Epoch: 80, train_loss: 0.0, test_loss: 1.2145 // +
Epoch: 81, train_loss: 0.0, test_loss: 1.2358 // +
Epoch: 82, train_loss: 0.0, test_loss: 1.1565 // -
Epoch: 83, train_loss: 0.0, test_loss: 1.0777 // -
Epoch: 84, train_loss: 0.0, test_loss: 1.1854 // +
Epoch: 85, train_loss: 0.0, test_loss: 1.1553 // -
Epoch: 86, train_loss: 0.0, test_loss: 1.1124 // -
Epoch: 87, train_loss: 0.0, test_loss: 1.094 // -
Epoch: 88, train_loss: 0.0013, test_loss: 1.2117 // +
Epoch: 89, train_loss: 0.0001, test_loss: 1.1872 // -
Epoch: 90, train_loss: 0.0, test_loss: 1.1481 // -
Epoch: 91, train_loss: 0.0, test_loss: 1.1924 // +
Epoch: 92, train_loss: 0.0, test_loss: 1.2567 // +
Epoch: 93, train_loss: 0.0, test_loss: 1.2277 // -
Epoch: 94, train_loss: 0.0, test_loss: 1.0857 // -
Epoch: 95, train_loss: 0.0, test_loss: 1.1796 // +
Epoch: 96, train_loss: 0.0, test_loss: 1.1804 // +
Epoch: 97, train_loss: 0.0, test_loss: 1.154 // -
Epoch: 98, train_loss: 0.0, test_loss: 1.2661 // +
Epoch: 99, train_loss: 0.0, test_loss: 1.0401 // -
Epoch: 100, train_loss: 0.0, test_loss: 1.242 // +
```

[2]:
```
import pandas as pd
df = pd.read_csv('resnet_epoch_wise.csv')
test_loss = {}
test_loss[3] = dict(zip(df['epoch'].values,df['3'].values))
test_loss[12] = dict(zip(df['epoch'].values,df['12'].values))
test_loss[48] = dict(zip(df['epoch'].values,df['48'].values))
```

[3]:
```
def movingaverage(interval, window_size):
    window= np.ones(int(window_size))/float(window_size)
    return np.convolve(interval, window, 'same')
```

```
y_av_3 = movingaverage(list(test_loss[3].values()), 5)
y_av_12 = movingaverage(list(test_loss[12].values()), 5)
y_av_48 = movingaverage(list(test_loss[48].values()), 5)
```

[4]:
```
#multiple line plot
plt.figure(figsize=(12,9))
plt.scatter(list(test_loss[3].keys()),list(test_loss[3].values()),marker =␣
↪'o',label='k=3',
            facecolor='yellow',color='skyblue',s=20,alpha=0.4)
plt.plot(list(test_loss[3].keys())[2:-2],y_av_3[2:-2],'bx-',marker = '.
↪',linestyle='-', color='skyblue',
         linewidth=2,label='mov.avg for k=3')
plt.scatter(list(test_loss[12].keys()),list(test_loss[12].values()),marker =␣
↪'o',label='k=12',
            facecolor='blue',color='purple',s=20,alpha=0.4)
plt.plot(list(test_loss[12].keys())[2:-2],y_av_12[2:-2],'bx-',marker = '.
↪',linestyle='-', color='purple',
         linewidth=2,label='mov.avg for k=12')
plt.scatter(list(test_loss[48].keys()),list(test_loss[48].values()),marker =␣
↪'o',label='k=48',
            facecolor='orange',color='tomato',s=20,alpha=0.4)
plt.plot(list(test_loss[48].keys())[2:-2],y_av_48[2:-2],'bx-',marker = '.
↪',linestyle='-', color='tomato',
         linewidth=2,label='mov.avg for k=48')
plt.xlabel('Epochs',fontsize=16)
plt.ylabel('Test Loss',fontsize=16)
plt.title('ResNet Epoch-Wise',fontsize=20)
plt.legend()
plt.grid()
plt.show()
```


ResNet Epoch-Wise
mov.avg for k=3
mov.avg for k=12
mov.avg for k=48
k=3
k=12
k=48
Test Loss
Epochs
2.25
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0
20
40
60
80
100