In [1]:
```
from keras.datasets import mnist
from keras import models
from keras import layers
from keras.layers.core import Dense, Dropout, Activation
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()
print("[Info] train data={:7,}".format(len(train_images)))
print("[Info] test  data={:7,}".format(len(test_images)))
```

```
D:\Anaconda3\lib\site-packages\h5py\__init__.py:36: FutureWarning: Conversion of the second argument of issubdtype from `float` to `np.floating` is deprecated. In future, it will be treated as `np.float64 == np.dtype(float).type`.
  from ._conv import register_converters as _register_converters
Using TensorFlow backend.

[Info] train data= 60,000
[Info] test  data= 10,000
```

In [2]:
```
train_images = train_images.reshape((60000, 28 * 28))
train_images = train_images.astype('float32') / 255

test_images = test_images.reshape((10000, 28 * 28))
test_images = test_images.astype('float32') / 255

from keras.utils import to_categorical

train_labels = to_categorical(train_labels)
test_labels = to_categorical(test_labels)
```

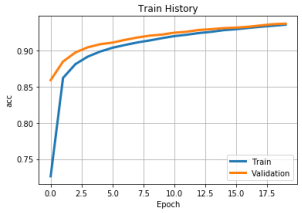
In [3]:
```
import matplotlib.pyplot as plt
%matplotlib inline
```

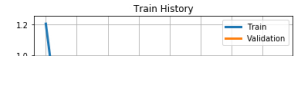
In [4]:
```
def show_train_history(train_history, train, validation, layer):
    plt.plot(train_history.history[train], linewidth=3)
    plt.plot(train_history.history[validation], linewidth=3)
    plt.title('Train History')
    plt.ylabel(train)
    plt.xlabel('Epoch')
    plt.legend(['Train', 'Validation'], loc='best')
    plt.grid(True)
    if train == 'acc':
        plt.savefig("hidden layers_acc_" + str(layer) + ".jpg")
    if train == 'loss':
        plt.savefig("hidden layers_loss_" + str(layer) + ".jpg")
    plt.show()
```

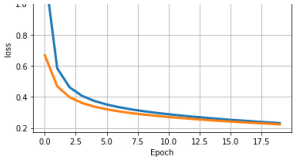
In [5]:
```
#1 Hidden Layers
network = models.Sequential()
network.add(layers.Dense(512, activation='relu', input_shape=(28 * 28,)))
network.add(layers.Dense(10, activation='softmax'))
network.compile(optimizer='sgd', loss='categorical_crossentropy', metrics=['accuracy'])
train_history_1 = network.fit(train_images, train_labels, validation_split=0.2, epochs=20, batch_size=128)
test_loss, test_acc = network.evaluate(test_images, test_labels)
print('test_loss:', test_loss)
print('test_acc:', test_acc)
#print(train_history.history)

print("[Info] Model summary:")
network.summary()
```

```
WARNING:tensorflow:From D:\Anaconda3\lib\site-packages\tensorflow\python\framework\op_def_library.py:263: colocate_with (from tensorflow.python.framework.ops) is deprecated and will be removed in a future version.
Instructions for updating:
Colocations handled automatically by placer.
WARNING:tensorflow:From D:\Anaconda3\lib\site-packages\tensorflow\python\ops\math_ops.py:3066: to_int32 (from tensorflow.python.ops.math_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.cast instead.
Train on 48000 samples, validate on 12000 samples
Epoch 1/20
48000/48000 [==============================] - 6s 133us/step - loss: 1.2045 - acc: 0.7261 - val_loss: 0.6678 - val_acc: 0.8592
Epoch 2/20
48000/48000 [==============================] - 2s 38us/step - loss: 0.5835 - acc: 0.8623 - val_loss: 0.4681 - val_acc: 0.8850
Epoch 3/20
48000/48000 [==============================] - 2s 38us/step - loss: 0.4614 - acc: 0.8814 - val_loss: 0.3976 - val_acc: 0.8976
Epoch 4/20
48000/48000 [==============================] - 2s 38us/step - loss: 0.4065 - acc: 0.8918 - val_loss: 0.3605 - val_acc: 0.9048
Epoch 5/20
48000/48000 [==============================] - 2s 41us/step - loss: 0.3735 - acc: 0.8988 - val_loss: 0.3370 - val_acc: 0.9090
Epoch 6/20
48000/48000 [==============================] - 2s 39us/step - loss: 0.3506 - acc: 0.9041 - val_loss: 0.3199 - val_acc: 0.9112
Epoch 7/20
48000/48000 [==============================] - 2s 41us/step - loss: 0.3332 - acc: 0.9079 - val_loss: 0.3062 - val_acc: 0.9150
Epoch 8/20
48000/48000 [==============================] - 2s 38us/step - loss: 0.3192 - acc: 0.9114 - val_loss: 0.2953 - val_acc: 0.9182
Epoch 9/20
48000/48000 [==============================] - 2s 38us/step - loss: 0.3073 - acc: 0.9142 - val_loss: 0.2855 - val_acc: 0.9209
Epoch 10/20
48000/48000 [==============================] - 2s 38us/step - loss: 0.2970 - acc: 0.9174 - val_loss: 0.2780 - val_acc: 0.9223
Epoch 11/20
48000/48000 [==============================] - 2s 41us/step - loss: 0.2880 - acc: 0.9203 - val_loss: 0.2699 - val_acc: 0.9250
Epoch 12/20
48000/48000 [==============================] - 2s 41us/step - loss: 0.2796 - acc: 0.9221 - val_loss: 0.2634 - val_acc: 0.9263
Epoch 13/20
48000/48000 [==============================] - 2s 40us/step - loss: 0.2721 - acc: 0.9245 - val_loss: 0.2574 - val_acc: 0.9287
Epoch 14/20
48000/48000 [==============================] - 2s 41us/step - loss: 0.2652 - acc: 0.9262 - val_loss: 0.2515 - val_acc: 0.9299
Epoch 15/20
48000/48000 [==============================] - 2s 41us/step - loss: 0.2587 - acc: 0.9285 - val_loss: 0.2459 - val_acc: 0.9313
Epoch 16/20
48000/48000 [==============================] - 2s 40us/step - loss: 0.2525 - acc: 0.9298 - val_loss: 0.2411 - val_acc: 0.9320
Epoch 17/20
48000/48000 [==============================] - 2s 42us/step - loss: 0.2469 - acc: 0.9315 - val_loss: 0.2361 - val_acc: 0.9332
Epoch 18/20
48000/48000 [==============================] - 2s 40us/step - loss: 0.2414 - acc: 0.9332 - val_loss: 0.2318 - val_acc: 0.9351
Epoch 19/20
48000/48000 [==============================] - 2s 40us/step - loss: 0.2361 - acc: 0.9346 - val_loss: 0.2277 - val_acc: 0.9367
Epoch 20/20
48000/48000 [==============================] - 2s 41us/step - loss: 0.2313 - acc: 0.9360 - val_loss: 0.2228 - val_acc: 0.9375
10000/10000 [==============================] - 0s 46us/step
test_loss: 0.22501645919084548
test_acc: 0.9366
[Info] Model summary:
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 512)               401920
_________________________________________________________________
dense_2 (Dense)              (None, 10)                5130
=================================================================
Total params: 407,050
Trainable params: 407,050
Non-trainable params: 0
_________________________________________________________________
```

In [6]:
```
layer = 1
show_train_history(train_history_1, 'acc', 'val_acc', layer)
show_train_history(train_history_1, 'loss', 'val_loss', layer)

scores_1 = network.evaluate(train_images, train_labels)
print()
print("[Info] Accuracy of testing data = {:2.1f}%".format(scores_1[1]*100.0))
```








```
60000/60000 [==============================] - 3s 49us/step

[Info] Accuracy of testing data = 93.7%
```

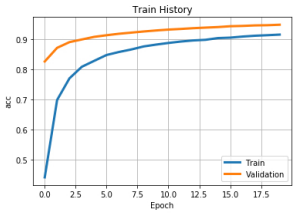
In [7]:

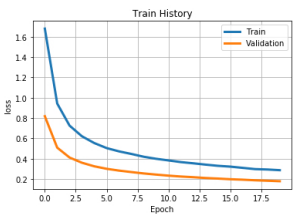
```
#2 Hidden Layers
network = models.Sequential()
network.add(layers.Dense(512, activation='relu', input_shape=(28 * 28,)))
network.add(layers.Dropout(0.5))
network.add(layers.Dense(128, activation='relu'))
network.add(layers.Dropout(0.5))
network.add(layers.Dense(10, activation='softmax'))
network.compile(optimizer='sgd', loss='categorical_crossentropy', metrics=['accuracy'])
train_history_2 = network.fit(train_images, train_labels, validation_split=0.2, epochs=20, batch_size=128)
test_loss, test_acc = network.evaluate(test_images, test_labels)
print('test_loss:', test_loss)
print('test_acc:', test_acc)
#print(train_history.history)

print("[Info] Model summary:")
network.summary()
```

```
WARNING:tensorflow:From D:\Anaconda3\lib\site-packages\keras\backend\tensorflow_backend.py:3445: calling dropout (from tensorfl
ow.python.ops.nn_ops) with keep_prob is deprecated and will be removed in a future version.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 - keep_prob`.
Train on 48000 samples, validate on 12000 samples
Epoch 1/20
48000/48000 [==============================] - 3s 56us/step - loss: 1.6794 - acc: 0.4414 - val_loss: 0.8179 - val_acc: 0.8267
Epoch 2/20
48000/48000 [==============================] - 2s 48us/step - loss: 0.9426 - acc: 0.6986 - val_loss: 0.5085 - val_acc: 0.8721
Epoch 3/20
48000/48000 [==============================] - 2s 48us/step - loss: 0.7252 - acc: 0.7706 - val_loss: 0.4111 - val_acc: 0.8911
Epoch 4/20
48000/48000 [==============================] - 2s 48us/step - loss: 0.6203 - acc: 0.8091 - val_loss: 0.3600 - val_acc: 0.8999
Epoch 5/20
48000/48000 [==============================] - 2s 47us/step - loss: 0.5536 - acc: 0.8288 - val_loss: 0.3247 - val_acc: 0.9086
Epoch 6/20
48000/48000 [==============================] - 2s 47us/step - loss: 0.5041 - acc: 0.8483 - val_loss: 0.3004 - val_acc: 0.9140
Epoch 7/20
48000/48000 [==============================] - 2s 49us/step - loss: 0.4717 - acc: 0.8582 - val_loss: 0.2826 - val_acc: 0.9189
Epoch 8/20
48000/48000 [==============================] - 2s 49us/step - loss: 0.4464 - acc: 0.8666 - val_loss: 0.2682 - val_acc: 0.9227
Epoch 9/20
48000/48000 [==============================] - 2s 46us/step - loss: 0.4187 - acc: 0.8767 - val_loss: 0.2545 - val_acc: 0.9265
Epoch 10/20
48000/48000 [==============================] - 2s 46us/step - loss: 0.3989 - acc: 0.8828 - val_loss: 0.2433 - val_acc: 0.9298
Epoch 11/20
48000/48000 [==============================] - 2s 45us/step - loss: 0.3828 - acc: 0.8883 - val_loss: 0.2330 - val_acc: 0.9327
Epoch 12/20
48000/48000 [==============================] - 2s 47us/step - loss: 0.3663 - acc: 0.8930 - val_loss: 0.2243 - val_acc: 0.9349
Epoch 13/20
48000/48000 [==============================] - 2s 46us/step - loss: 0.3543 - acc: 0.8966 - val_loss: 0.2177 - val_acc: 0.9374
Epoch 14/20
48000/48000 [==============================] - 2s 47us/step - loss: 0.3413 - acc: 0.8988 - val_loss: 0.2099 - val_acc: 0.9393
Epoch 15/20
48000/48000 [==============================] - 2s 46us/step - loss: 0.3297 - acc: 0.9045 - val_loss: 0.2043 - val_acc: 0.9412
Epoch 16/20
48000/48000 [==============================] - 2s 46us/step - loss: 0.3216 - acc: 0.9060 - val_loss: 0.1978 - val_acc: 0.9439
Epoch 17/20
48000/48000 [==============================] - 2s 50us/step - loss: 0.3096 - acc: 0.9096 - val_loss: 0.1933 - val_acc: 0.9450
Epoch 18/20
48000/48000 [==============================] - 2s 50us/step - loss: 0.2973 - acc: 0.9123 - val_loss: 0.1874 - val_acc: 0.9467
Epoch 19/20
48000/48000 [==============================] - 2s 45us/step - loss: 0.2934 - acc: 0.9141 - val_loss: 0.1830 - val_acc: 0.9473
Epoch 20/20
48000/48000 [==============================] - 2s 43us/step - loss: 0.2867 - acc: 0.9161 - val_loss: 0.1781 - val_acc: 0.9489
10000/10000 [==============================] - 0s 46us/step
test_loss: 0.1774854830875993
test_acc: 0.9465
[Info] Model summary:
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_3 (Dense)              (None, 512)               401920
_________________________________________________________________
dropout_1 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_4 (Dense)              (None, 128)               65664
_________________________________________________________________
dropout_2 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_5 (Dense)              (None, 10)                1290
=================================================================
Total params: 468,874
Trainable params: 468,874
Non-trainable params: 0
_________________________________________________________________
```

In [8]:

```
layer = 2
show_train_history(train_history_2, 'acc', 'val_acc', layer)
show_train_history(train_history_2, 'loss', 'val_loss', layer)

scores_2 = network.evaluate(train_images, train_labels)
print()
print("[Info] Accuracy of testing data = {:2.1f}%".format(scores_2[1]*100.0))
```






```
60000/60000 [==============================] - 3s 51us/step

[Info] Accuracy of testing data = 94.7%
```

In [9]:

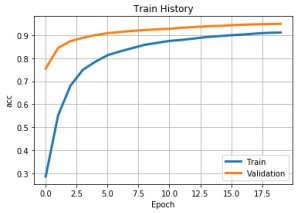
```
#3 Hidden Layers
network = models.Sequential()
network.add(layers.Dense(512, activation='relu', input_shape=(28 * 28,)))
network.add(Dropout(0.5))
network.add(layers.Dense(256, activation='relu'))
network.add(Dropout(0.5))
network.add(layers.Dense(128, activation='relu'))
network.add(Dropout(0.5))
network.add(layers.Dense(10, activation='softmax'))
network.compile(optimizer='sgd', loss='categorical_crossentropy', metrics=['accuracy'])
train_history_3 = network.fit(train_images, train_labels, validation_split=0.2, epochs=20, batch_size=128)
test_loss, test_acc = network.evaluate(test_images, test_labels)
print('test_loss:', test_loss)
print('test_acc:', test_acc)
#print(train_history.history)
```

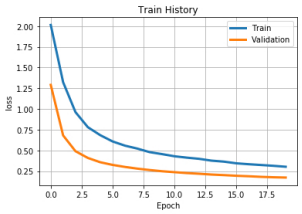
```
print("[Info] Model summary:")
network.summary()
```

```
Train on 48000 samples, validate on 12000 samples
Epoch 1/20
48000/48000 [==============================] - 3s 61us/step - loss: 2.0142 - acc: 0.2865 - val_loss: 1.2907 - val_acc: 0.7552
Epoch 2/20
48000/48000 [==============================] - 2s 52us/step - loss: 1.3230 - acc: 0.5519 - val_loss: 0.6806 - val_acc: 0.8445
Epoch 3/20
48000/48000 [==============================] - 3s 54us/step - loss: 0.9616 - acc: 0.6805 - val_loss: 0.4900 - val_acc: 0.8739
Epoch 4/20
48000/48000 [==============================] - 2s 49us/step - loss: 0.7801 - acc: 0.7492 - val_loss: 0.4087 - val_acc: 0.8885
Epoch 5/20
48000/48000 [==============================] - 2s 48us/step - loss: 0.6838 - acc: 0.7840 - val_loss: 0.3569 - val_acc: 0.8999
Epoch 6/20
48000/48000 [==============================] - 2s 50us/step - loss: 0.6078 - acc: 0.8130 - val_loss: 0.3247 - val_acc: 0.9087
Epoch 7/20
48000/48000 [==============================] - 2s 51us/step - loss: 0.5571 - acc: 0.8291 - val_loss: 0.3002 - val_acc: 0.9135
Epoch 8/20
48000/48000 [==============================] - 2s 50us/step - loss: 0.5217 - acc: 0.8436 - val_loss: 0.2796 - val_acc: 0.9183
Epoch 9/20
48000/48000 [==============================] - 3s 57us/step - loss: 0.4786 - acc: 0.8579 - val_loss: 0.2628 - val_acc: 0.9218
Epoch 10/20
48000/48000 [==============================] - 2s 51us/step - loss: 0.4549 - acc: 0.8663 - val_loss: 0.2481 - val_acc: 0.9255
Epoch 11/20
48000/48000 [==============================] - 2s 52us/step - loss: 0.4291 - acc: 0.8749 - val_loss: 0.2368 - val_acc: 0.9275
Epoch 12/20
48000/48000 [==============================] - 2s 51us/step - loss: 0.4122 - acc: 0.8792 - val_loss: 0.2261 - val_acc: 0.9319
Epoch 13/20
48000/48000 [==============================] - 2s 51us/step - loss: 0.3980 - acc: 0.8852 - val_loss: 0.2179 - val_acc: 0.9351
Epoch 14/20
48000/48000 [==============================] - 3s 53us/step - loss: 0.3760 - acc: 0.8917 - val_loss: 0.2081 - val_acc: 0.9384
Epoch 15/20
48000/48000 [==============================] - 2s 51us/step - loss: 0.3642 - acc: 0.8956 - val_loss: 0.2014 - val_acc: 0.9398
Epoch 16/20
48000/48000 [==============================] - 2s 50us/step - loss: 0.3440 - acc: 0.8995 - val_loss: 0.1935 - val_acc: 0.9430
Epoch 17/20
48000/48000 [==============================] - 2s 51us/step - loss: 0.3329 - acc: 0.9027 - val_loss: 0.1877 - val_acc: 0.9452
Epoch 18/20
48000/48000 [==============================] - 2s 50us/step - loss: 0.3235 - acc: 0.9070 - val_loss: 0.1806 - val_acc: 0.9466
Epoch 19/20
48000/48000 [==============================] - 2s 50us/step - loss: 0.3137 - acc: 0.9100 - val_loss: 0.1754 - val_acc: 0.9479
Epoch 20/20
48000/48000 [==============================] - 2s 49us/step - loss: 0.3025 - acc: 0.9113 - val_loss: 0.1720 - val_acc: 0.9490
10000/10000 [==============================] - 1s 52us/step
test_loss: 0.17379450362809
test_acc: 0.9469
[Info] Model summary:
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_6 (Dense)              (None, 512)               401920
_________________________________________________________________
dropout_3 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_7 (Dense)              (None, 256)               131328
_________________________________________________________________
dropout_4 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_8 (Dense)              (None, 128)               32896
_________________________________________________________________
dropout_5 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_9 (Dense)              (None, 10)                1290
=================================================================
Total params: 567,434
Trainable params: 567,434
Non-trainable params: 0
_________________________________________________________________
```

In [10]:

```
layer = 3
show_train_history(train_history_3, 'acc', 'val_acc', layer)
show_train_history(train_history_3, 'loss', 'val_loss', layer)

scores_3 = network.evaluate(train_images, train_labels)
print()
print("[Info] Accuracy of testing data = {:2.1f}%".format(scores_3[1]*100.0))
```

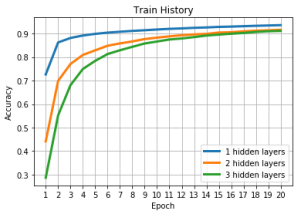





```
60000/60000 [==============================] - 3s 49us/step

[Info] Accuracy of testing data = 94.9%
```

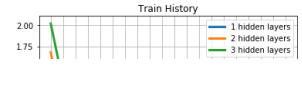
In [11]:

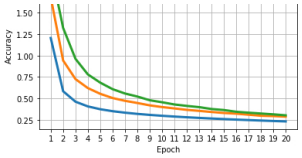
```
plt.plot(train_history_1.history['acc'], linewidth=3)
plt.plot(train_history_2.history['acc'], linewidth=3)
plt.plot(train_history_3.history['acc'], linewidth=3)
plt.title('Train History')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.xticks([0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19], ['1','2','3','4','5','6','7','8','9','10','11','12','13','14'
plt.legend(['1 hidden layers','2 hidden layers','3 hidden layers'], loc='best')
plt.grid(True)
plt.savefig('hidden_layers_acc_all.jpg',dpi=300)
```



In [12]:

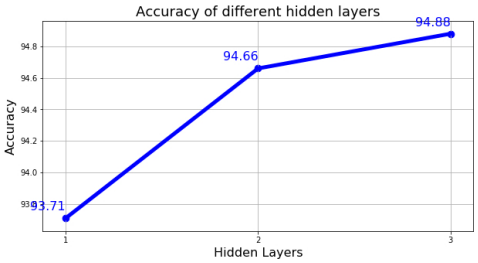
```
plt.plot(train_history_1.history['loss'], linewidth=3)
plt.plot(train_history_2.history['loss'], linewidth=3)
plt.plot(train_history_3.history['loss'], linewidth=3)
plt.title('Train History')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.xticks([0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19], ['1','2','3','4','5','6','7','8','9','10','11','12','13','14'
plt.legend(['1 hidden layers','2 hidden layers','3 hidden layers'], loc='best')
plt.grid(True)
plt.savefig('hidden_layers_loss_all.jpg',dpi=300)
```





In [13]:

```
acc = [round(scores_1[1]*100,2),round(scores_2[1]*100,2),round(scores_3[1]*100,2)]
x = ['1','2','3']
plt.figure(figsize=(10,5))
plt.plot(x,acc,'b',lw=5)
plt.scatter(x, acc, s = 75,color='b')
plt.title('Accuracy of different hidden layers', fontsize='18')
plt.xlabel('Hidden Layers',fontsize=16)
plt.ylabel('Accuracy',fontsize=16)
plt.grid(True)
for x,y in enumerate(acc):
    plt.text(x,y+0.05,'%s' %y, ha='right', color='b',fontsize=16)
plt.savefig('Accuracy of different hidden layers.jpg',dpi=300)
```



In [ ]: