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)
```

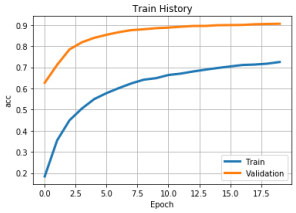
In [3]:
```
import matplotlib.pyplot as plt
%matplotlib inline
```

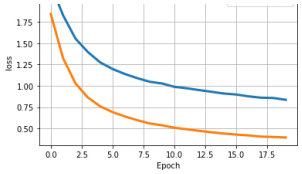
In [4]:
```
def show_train_history(train_history, train, validation, node):
    plt.plot(train_history.history[train], linewidth=3)
    plt.plot(train_history.history[validation], linewidth=3)
    plt.title('Train History')
    plt.ylabel(train)
    plt.xlabel('Epoch')
    plt.legend(['Train', 'Validation'],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

In [12]:
```
#32 nodes
network = models.Sequential()
network.add(layers.Dense(32, activation='relu', input_shape=(28 * 28,)))
network.add(Dropout(0.5))
network.add(layers.Dense(32, 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_32 = 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 [==============================] - 2s 44us/step - loss: 2.1841 - acc: 0.1841 - val_loss: 1.8440 - val_acc: 0.6265
Epoch 2/20
48000/48000 [==============================] - 2s 37us/step - loss: 1.8265 - acc: 0.3540 - val_loss: 1.3228 - val_acc: 0.7111
Epoch 3/20
48000/48000 [==============================] - 2s 35us/step - loss: 1.5544 - acc: 0.4490 - val_loss: 1.0291 - val_acc: 0.7841
Epoch 4/20
48000/48000 [==============================] - 2s 35us/step - loss: 1.3969 - acc: 0.5045 - val_loss: 0.8635 - val_acc: 0.8177
Epoch 5/20
48000/48000 [==============================] - 2s 33us/step - loss: 1.2747 - acc: 0.5494 - val_loss: 0.7595 - val_acc: 0.8388
Epoch 6/20
48000/48000 [==============================] - 2s 39us/step - loss: 1.1977 - acc: 0.5777 - val_loss: 0.6921 - val_acc: 0.8534
Epoch 7/20
48000/48000 [==============================] - 2s 32us/step - loss: 1.1387 - acc: 0.6016 - val_loss: 0.6408 - val_acc: 0.8657
Epoch 8/20
48000/48000 [==============================] - 2s 35us/step - loss: 1.0903 - acc: 0.6234 - val_loss: 0.5974 - val_acc: 0.8752
Epoch 9/20
48000/48000 [==============================] - 2s 35us/step - loss: 1.0483 - acc: 0.6410 - val_loss: 0.5592 - val_acc: 0.8794
Epoch 10/20
48000/48000 [==============================] - 2s 31us/step - loss: 1.0278 - acc: 0.6483 - val_loss: 0.5374 - val_acc: 0.8844
Epoch 11/20
48000/48000 [==============================] - 2s 33us/step - loss: 0.9886 - acc: 0.6632 - val_loss: 0.5092 - val_acc: 0.8874
Epoch 12/20
48000/48000 [==============================] - 2s 32us/step - loss: 0.9718 - acc: 0.6699 - val_loss: 0.4910 - val_acc: 0.8914
Epoch 13/20
48000/48000 [==============================] - 2s 35us/step - loss: 0.9504 - acc: 0.6799 - val_loss: 0.4728 - val_acc: 0.8950
Epoch 14/20
48000/48000 [==============================] - 2s 32us/step - loss: 0.9304 - acc: 0.6886 - val_loss: 0.4556 - val_acc: 0.8953
Epoch 15/20
48000/48000 [==============================] - 2s 36us/step - loss: 0.9099 - acc: 0.6965 - val_loss: 0.4411 - val_acc: 0.8985
Epoch 16/20
48000/48000 [==============================] - 2s 33us/step - loss: 0.8987 - acc: 0.7037 - val_loss: 0.4291 - val_acc: 0.8992
Epoch 17/20
48000/48000 [==============================] - 2s 36us/step - loss: 0.8769 - acc: 0.7106 - val_loss: 0.4187 - val_acc: 0.8999
Epoch 18/20
48000/48000 [==============================] - 2s 41us/step - loss: 0.8613 - acc: 0.7127 - val_loss: 0.4052 - val_acc: 0.9031
Epoch 19/20
48000/48000 [==============================] - 2s 39us/step - loss: 0.8577 - acc: 0.7169 - val_loss: 0.4006 - val_acc: 0.9045
Epoch 20/20
48000/48000 [==============================] - 2s 41us/step - loss: 0.8369 - acc: 0.7247 - val_loss: 0.3945 - val_acc: 0.9053
10000/10000 [==============================] - 1s 56us/step
test_loss: 0.3999697699069977
test_acc: 0.9021
[Info] Model summary:
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_10 (Dense)             (None, 32)                25120
_________________________________________________________________
dropout_7 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_11 (Dense)             (None, 32)                1056
_________________________________________________________________
dropout_8 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_12 (Dense)             (None, 10)                330
=================================================================
Total params: 26,506
Trainable params: 26,506
Non-trainable params: 0
_________________________________________________________________
```

In [13]:
```
node = 32
show_train_history(train_history_32, 'acc', 'val_acc', node)
show_train_history(train_history_32, 'loss', 'val_loss', node)

scores_32 = network.evaluate(train_images, train_labels)
print()
print("[Info] Accuracy of testing data = {:2.1f}%".format(scores_32[1]*100.0))
```






```
60000/60000 [==============================] - 3s 44us/step

[Info] Accuracy of testing data = 90.1%
```

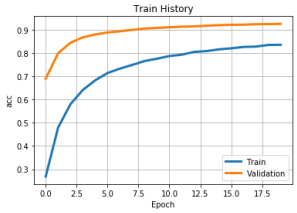
In [5]:

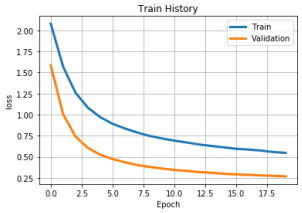
```
#64 nodes
network = models.Sequential()
network.add(layers.Dense(64, activation='relu', input_shape=(28 * 28,)))
network.add(Dropout(0.5))
network.add(layers.Dense(64, activation='relu'))
network.add(Dropout(0.5))
network.add(layers.Dense(10, activation='softmax'))
network.compile(optimizer='sgd', loss='categorical_crossentropy', metrics=['accuracy'])
train_history_64 = 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 t
ensorflow.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\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`.
WARNING:tensorflow:From D:\Anaconda3\lib\site-packages\tensorflow\python\ops\math_ops.py:3066: to_int32 (from tensorflow.pytho
n.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 [==============================] - 5s 98us/step - loss: 2.0791 - acc: 0.2692 - val_loss: 1.5866 - val_acc: 0.6893
Epoch 2/20
48000/48000 [==============================] - 2s 34us/step - loss: 1.5692 - acc: 0.4791 - val_loss: 1.0052 - val_acc: 0.7987
Epoch 3/20
48000/48000 [==============================] - 2s 34us/step - loss: 1.2605 - acc: 0.5797 - val_loss: 0.7407 - val_acc: 0.8436
Epoch 4/20
48000/48000 [==============================] - 2s 36us/step - loss: 1.0826 - acc: 0.6409 - val_loss: 0.6054 - val_acc: 0.8667
Epoch 5/20
48000/48000 [==============================] - 2s 33us/step - loss: 0.9704 - acc: 0.6820 - val_loss: 0.5247 - val_acc: 0.8791
Epoch 6/20
48000/48000 [==============================] - 2s 36us/step - loss: 0.8908 - acc: 0.7136 - val_loss: 0.4736 - val_acc: 0.8877
Epoch 7/20
48000/48000 [==============================] - 2s 36us/step - loss: 0.8356 - acc: 0.7324 - val_loss: 0.4347 - val_acc: 0.8929
Epoch 8/20
48000/48000 [==============================] - 2s 36us/step - loss: 0.7878 - acc: 0.7484 - val_loss: 0.4014 - val_acc: 0.8991
Epoch 9/20
48000/48000 [==============================] - 2s 34us/step - loss: 0.7471 - acc: 0.7652 - val_loss: 0.3789 - val_acc: 0.9048
Epoch 10/20
48000/48000 [==============================] - 2s 35us/step - loss: 0.7174 - acc: 0.7750 - val_loss: 0.3607 - val_acc: 0.9073
Epoch 11/20
48000/48000 [==============================] - 2s 32us/step - loss: 0.6913 - acc: 0.7864 - val_loss: 0.3432 - val_acc: 0.9102
Epoch 12/20
48000/48000 [==============================] - 2s 35us/step - loss: 0.6695 - acc: 0.7922 - val_loss: 0.3319 - val_acc: 0.9126
Epoch 13/20
48000/48000 [==============================] - 2s 35us/step - loss: 0.6457 - acc: 0.8042 - val_loss: 0.3180 - val_acc: 0.9137
Epoch 14/20
48000/48000 [==============================] - 2s 35us/step - loss: 0.6280 - acc: 0.8077 - val_loss: 0.3107 - val_acc: 0.9167
Epoch 15/20
48000/48000 [==============================] - 2s 35us/step - loss: 0.6115 - acc: 0.8151 - val_loss: 0.2983 - val_acc: 0.9193
Epoch 16/20
48000/48000 [==============================] - 2s 36us/step - loss: 0.5946 - acc: 0.8196 - val_loss: 0.2897 - val_acc: 0.9205
Epoch 17/20
48000/48000 [==============================] - 2s 39us/step - loss: 0.5855 - acc: 0.8257 - val_loss: 0.2857 - val_acc: 0.9213
Epoch 18/20
48000/48000 [==============================] - 2s 39us/step - loss: 0.5734 - acc: 0.8271 - val_loss: 0.2786 - val_acc: 0.9231
Epoch 19/20
48000/48000 [==============================] - 2s 38us/step - loss: 0.5565 - acc: 0.8341 - val_loss: 0.2739 - val_acc: 0.9237
Epoch 20/20
48000/48000 [==============================] - 2s 38us/step - loss: 0.5461 - acc: 0.8348 - val_loss: 0.2677 - val_acc: 0.9247
10000/10000 [==============================] - 1s 57us/step
test_loss: 0.275950950488448l4
test_acc: 0.9228
[Info] Model summary:
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 64)                50240
_________________________________________________________________
dropout_1 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_2 (Dense)              (None, 64)                4160
_________________________________________________________________
dropout_2 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_3 (Dense)              (None, 10)                650
=================================================================
Total params: 55,050
Trainable params: 55,050
Non-trainable params: 0
_________________________________________________________________
```

In [7]:

```
node = 64
show_train_history(train_history_64, 'acc', 'val_acc', node)
show_train_history(train_history_64, 'loss', 'val_loss', node)

scores_64 = network.evaluate(train_images, train_labels)
print()
print("[Info] Accuracy of testing data = {:2.1f}%".format(scores_64[1]*100.0))
```






```
60000/60000 [==============================] - 3s 45us/step

[Info] Accuracy of testing data = 92.1%
```

In [8]:

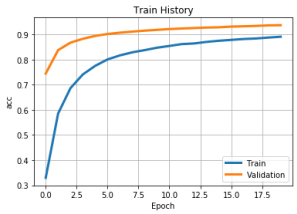
```
#128 nodes
network = models.Sequential()
network.add(layers.Dense(128, activation='relu', input_shape=(28 * 28,)))
network.add(Dropout(0.5))
network.add(layers.Dense(128, activation='relu'))
network.add(Dropout(0.5))
```

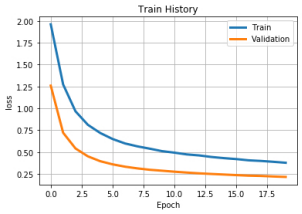
```
network.add(layers.Dense(10, activation='softmax'))
network.compile(optimizer='sgd', loss='categorical_crossentropy', metrics=['accuracy'])
train_history_128 = network.fit(train_images, train_labels, validation_split=0.2, epochs=20, batch_size=128)
test_loss, test_acc = network.evaluate(test_images, 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 [==============================] - 2s 43us/step - loss: 1.9604 - acc: 0.3298 - val_loss: 1.2600 - val_acc: 0.7442
Epoch 2/20
48000/48000 [==============================] - 2s 36us/step - loss: 1.2740 - acc: 0.5852 - val_loss: 0.7222 - val_acc: 0.8373
Epoch 3/20
48000/48000 [==============================] - 2s 37us/step - loss: 0.9668 - acc: 0.6861 - val_loss: 0.5425 - val_acc: 0.8671
Epoch 4/20
48000/48000 [==============================] - 2s 36us/step - loss: 0.8127 - acc: 0.7407 - val_loss: 0.4525 - val_acc: 0.8823
Epoch 5/20
48000/48000 [==============================] - 2s 35us/step - loss: 0.7192 - acc: 0.7748 - val_loss: 0.3968 - val_acc: 0.8938
Epoch 6/20
48000/48000 [==============================] - 2s 35us/step - loss: 0.6504 - acc: 0.8004 - val_loss: 0.3610 - val_acc: 0.9013
Epoch 7/20
48000/48000 [==============================] - 2s 37us/step - loss: 0.5998 - acc: 0.8164 - val_loss: 0.3343 - val_acc: 0.9067
Epoch 8/20
48000/48000 [==============================] - 2s 35us/step - loss: 0.5665 - acc: 0.8285 - val_loss: 0.3147 - val_acc: 0.9108
Epoch 9/20
48000/48000 [==============================] - 2s 36us/step - loss: 0.5400 - acc: 0.8373 - val_loss: 0.2989 - val_acc: 0.9147
Epoch 10/20
48000/48000 [==============================] - 2s 35us/step - loss: 0.5117 - acc: 0.8469 - val_loss: 0.2882 - val_acc: 0.9178
Epoch 11/20
48000/48000 [==============================] - 2s 36us/step - loss: 0.4943 - acc: 0.8539 - val_loss: 0.2767 - val_acc: 0.9211
Epoch 12/20
48000/48000 [==============================] - 2s 36us/step - loss: 0.4742 - acc: 0.8612 - val_loss: 0.2671 - val_acc: 0.9232
Epoch 13/20
48000/48000 [==============================] - 2s 36us/step - loss: 0.4628 - acc: 0.8638 - val_loss: 0.2581 - val_acc: 0.9251
Epoch 14/20
48000/48000 [==============================] - 2s 37us/step - loss: 0.4451 - acc: 0.8700 - val_loss: 0.2510 - val_acc: 0.9270
Epoch 15/20
48000/48000 [==============================] - 2s 36us/step - loss: 0.4319 - acc: 0.8745 - val_loss: 0.2446 - val_acc: 0.9281
Epoch 16/20
48000/48000 [==============================] - 2s 38us/step - loss: 0.4210 - acc: 0.8780 - val_loss: 0.2380 - val_acc: 0.9307
Epoch 17/20
48000/48000 [==============================] - 2s 41us/step - loss: 0.4069 - acc: 0.8815 - val_loss: 0.2317 - val_acc: 0.9320
Epoch 18/20
48000/48000 [==============================] - 2s 43us/step - loss: 0.3997 - acc: 0.8835 - val_loss: 0.2284 - val_acc: 0.9335
Epoch 19/20
48000/48000 [==============================] - 2s 39us/step - loss: 0.3899 - acc: 0.8873 - val_loss: 0.2220 - val_acc: 0.9357
Epoch 20/20
48000/48000 [==============================] - 2s 39us/step - loss: 0.3789 - acc: 0.8905 - val_loss: 0.2172 - val_acc: 0.9363
10000/10000 [==============================] - 1s 54us/step
test_loss: 0.21714475657641888
test_acc: 0.9363
[Info] Model summary:
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_4 (Dense)              (None, 128)               100480
_________________________________________________________________
dropout_3 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_5 (Dense)              (None, 128)               16512
_________________________________________________________________
dropout_4 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_6 (Dense)              (None, 10)                1290
=================================================================
Total params: 118,282
Trainable params: 118,282
Non-trainable params: 0
_________________________________________________________________
```

In [9]:

```
node = 128
show_train_history(train_history_128, 'acc', 'val_acc', node)
show_train_history(train_history_128, 'loss', 'val_loss', node)

scores_128 = network.evaluate(train_images, train_labels)
print()
print("[Info] Accuracy of testing data = {:2.1f}%".format(scores_128[1]*100.0))
```






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

[Info] Accuracy of testing data = 93.5%
```

In [10]:

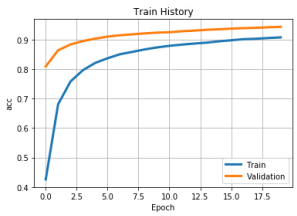
```
#256 nodes
network = models.Sequential()
network.add(layers.Dense(256, activation='relu', input_shape=(28 * 28,)))
network.add(Dropout(0.5))
network.add(layers.Dense(256, activation='relu'))
network.add(Dropout(0.5))
network.add(layers.Dense(10, activation='softmax'))
network.compile(optimizer='sgd', loss='categorical_crossentropy', metrics=['accuracy'])
train_history_256 = 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()
```

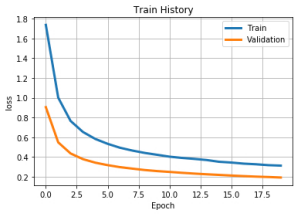
```
Train on 48000 samples, validate on 12000 samples
Epoch 1/20
48000/48000 [==============================] - 3s 55us/step - loss: 1.7380 - acc: 0.4256 - val_loss: 0.9051 - val_acc: 0.8098
Epoch 2/20
48000/48000 [==============================] - 2s 45us/step - loss: 1.0018 - acc: 0.6810 - val_loss: 0.5488 - val_acc: 0.8645
Epoch 3/20
48000/48000 [==============================] - 2s 49us/step - loss: 0.7658 - acc: 0.7582 - val_loss: 0.4365 - val_acc: 0.8844
Epoch 4/20
48000/48000 [==============================] - 2s 48us/step - loss: 0.6540 - acc: 0.7967 - val_loss: 0.3788 - val_acc: 0.8958
Epoch 5/20
48000/48000 [==============================] - 2s 46us/step - loss: 0.5832 - acc: 0.8216 - val_loss: 0.3430 - val_acc: 0.9042
Epoch 6/20
48000/48000 [==============================] - 2s 43us/step - loss: 0.5339 - acc: 0.8373 - val_loss: 0.3180 - val_acc: 0.9110
Epoch 7/20
48000/48000 [==============================] - 2s 44us/step - loss: 0.4941 - acc: 0.8511 - val_loss: 0.2977 - val_acc: 0.9157
Epoch 8/20
48000/48000 [==============================] - 2s 44us/step - loss: 0.4656 - acc: 0.8591 - val_loss: 0.2826 - val_acc: 0.9188
Epoch 9/20
48000/48000 [==============================] - 2s 43us/step - loss: 0.4417 - acc: 0.8675 - val_loss: 0.2685 - val_acc: 0.9219
Epoch 10/20
48000/48000 [==============================] - 2s 45us/step - loss: 0.4228 - acc: 0.8743 - val_loss: 0.2578 - val_acc: 0.9247
Epoch 11/20
48000/48000 [==============================] - 2s 41us/step - loss: 0.4039 - acc: 0.8798 - val_loss: 0.2494 - val_acc: 0.9258
Epoch 12/20
48000/48000 [==============================] - 2s 43us/step - loss: 0.3912 - acc: 0.8838 - val_loss: 0.2405 - val_acc: 0.9293
Epoch 13/20
```

```
48000/48000 [==============================] - 2s 44us/step - loss: 0.3812 - acc: 0.8874 - val_loss: 0.2326 - val_acc: 0.9313
Epoch 14/20
48000/48000 [==============================] - 2s 43us/step - loss: 0.3699 - acc: 0.8905 - val_loss: 0.2255 - val_acc: 0.9343
Epoch 15/20
48000/48000 [==============================] - 2s 45us/step - loss: 0.3523 - acc: 0.8950 - val_loss: 0.2193 - val_acc: 0.9359
Epoch 16/20
48000/48000 [==============================] - 2s 43us/step - loss: 0.3443 - acc: 0.8986 - val_loss: 0.2128 - val_acc: 0.9382
Epoch 17/20
48000/48000 [==============================] - 2s 45us/step - loss: 0.3333 - acc: 0.9024 - val_loss: 0.2069 - val_acc: 0.9398
Epoch 18/20
48000/48000 [==============================] - 2s 44us/step - loss: 0.3269 - acc: 0.9038 - val_loss: 0.2020 - val_acc: 0.9409
Epoch 19/20
48000/48000 [==============================] - 2s 44us/step - loss: 0.3176 - acc: 0.9063 - val_loss: 0.1979 - val_acc: 0.9427
Epoch 20/20
48000/48000 [==============================] - 2s 42us/step - loss: 0.3131 - acc: 0.9084 - val_loss: 0.1928 - val_acc: 0.9444
10000/10000 [==============================] - 0s 49us/step
test_loss: 0.19413354705944658
test_acc: 0.9418
[Info] Model summary:
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_7 (Dense)              (None, 256)               200960
_________________________________________________________________
dropout_5 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_8 (Dense)              (None, 256)               65792
_________________________________________________________________
dropout_6 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_9 (Dense)              (None, 10)                2570
=================================================================
Total params: 269,322
Trainable params: 269,322
Non-trainable params: 0
_________________________________________________________________
```

In [11]:

```
node = 256
show_train_history(train_history_256, 'acc', 'val_acc', node)
show_train_history(train_history_256, 'loss', 'val_loss', node)

scores_256 = network.evaluate(train_images, train_labels)
print()
print("[Info] Accuracy of testing data = {:2.1f}%".format(scores_256[1]*100.0))
```





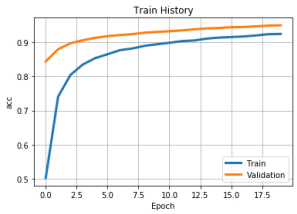

```
60000/60000 [==============================] - 3s 48us/step

[Info] Accuracy of testing data = 94.2%
```

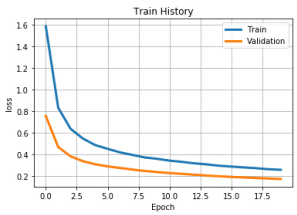
In [14]:

```
#512 nodes
network = models.Sequential()
network.add(layers.Dense(512, activation='relu', input_shape=(28 * 28,)))
network.add(Dropout(0.5))
network.add(layers.Dense(512, 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_512 = 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 62us/step - loss: 1.5858 - acc: 0.5028 - val_loss: 0.7561 - val_acc: 0.8438
Epoch 2/20
48000/48000 [==============================] - 3s 53us/step - loss: 0.8331 - acc: 0.7407 - val_loss: 0.4692 - val_acc: 0.8793
Epoch 3/20
48000/48000 [==============================] - 3s 53us/step - loss: 0.6380 - acc: 0.8041 - val_loss: 0.3830 - val_acc: 0.8968
Epoch 4/20
48000/48000 [==============================] - 2s 50us/step - loss: 0.5473 - acc: 0.8347 - val_loss: 0.3377 - val_acc: 0.9058
Epoch 5/20
48000/48000 [==============================] - 2s 51us/step - loss: 0.4867 - acc: 0.8533 - val_loss: 0.3092 - val_acc: 0.9127
Epoch 6/20
48000/48000 [==============================] - 2s 51us/step - loss: 0.4524 - acc: 0.8648 - val_loss: 0.2898 - val_acc: 0.9177
Epoch 7/20
48000/48000 [==============================] - 2s 51us/step - loss: 0.4195 - acc: 0.8766 - val_loss: 0.2751 - val_acc: 0.9209
Epoch 8/20
48000/48000 [==============================] - 2s 51us/step - loss: 0.3963 - acc: 0.8816 - val_loss: 0.2609 - val_acc: 0.9235
Epoch 9/20
48000/48000 [==============================] - 3s 52us/step - loss: 0.3726 - acc: 0.8894 - val_loss: 0.2471 - val_acc: 0.9278
Epoch 10/20
48000/48000 [==============================] - 2s 50us/step - loss: 0.3599 - acc: 0.8940 - val_loss: 0.2373 - val_acc: 0.9301
Epoch 11/20
48000/48000 [==============================] - 3s 57us/step - loss: 0.3430 - acc: 0.8983 - val_loss: 0.2278 - val_acc: 0.9320
Epoch 12/20
48000/48000 [==============================] - 3s 62us/step - loss: 0.3316 - acc: 0.9031 - val_loss: 0.2208 - val_acc: 0.9345
Epoch 13/20
48000/48000 [==============================] - 3s 59us/step - loss: 0.3185 - acc: 0.9050 - val_loss: 0.2127 - val_acc: 0.9376
Epoch 14/20
48000/48000 [==============================] - 3s 55us/step - loss: 0.3083 - acc: 0.9103 - val_loss: 0.2049 - val_acc: 0.9402
Epoch 15/20
48000/48000 [==============================] - 3s 59us/step - loss: 0.2959 - acc: 0.9134 - val_loss: 0.1987 - val_acc: 0.9414
Epoch 16/20
48000/48000 [==============================] - 3s 57us/step - loss: 0.2882 - acc: 0.9150 - val_loss: 0.1925 - val_acc: 0.9439
Epoch 17/20
48000/48000 [==============================] - 3s 53us/step - loss: 0.2804 - acc: 0.9168 - val_loss: 0.1875 - val_acc: 0.9445
Epoch 18/20
48000/48000 [==============================] - 3s 57us/step - loss: 0.2731 - acc: 0.9196 - val_loss: 0.1823 - val_acc: 0.9463
Epoch 19/20
48000/48000 [==============================] - 3s 58us/step - loss: 0.2637 - acc: 0.9234 - val_loss: 0.1776 - val_acc: 0.9485
Epoch 20/20
48000/48000 [==============================] - 3s 61us/step - loss: 0.2584 - acc: 0.9242 - val_loss: 0.1730 - val_acc: 0.9492
10000/10000 [==============================] - 1s 63us/step
test_loss: 0.17242722240537406
test_acc: 0.9478
[Info] Model summary:
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_13 (Dense)             (None, 512)               401920
_________________________________________________________________
dropout_9 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_14 (Dense)             (None, 512)               262656
_________________________________________________________________
dropout_10 (Dropout)         (None, 512)               0
_________________________________________________________________
dense_15 (Dense)             (None, 10)                5130
=================================================================
Total params: 669,706
Trainable params: 669,706
Non-trainable params: 0
_________________________________________________________________
```

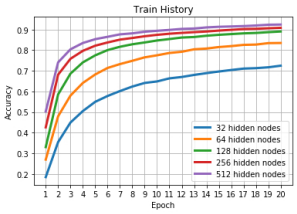
In [15]:
```
node = 512
show_train_history(train_history_512, 'acc', 'val_acc', node)
show_train_history(train_history_512, 'loss', 'val_loss', node)

scores_512 = network.evaluate(train_images, train_labels)
print()
print("[Info] Accuracy of testing data = {:2.1f}%".format(scores_512[1]*100.0))
```



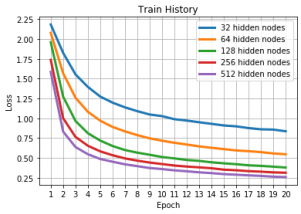



```
60000/60000 [==============================] - 4s 60us/step

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

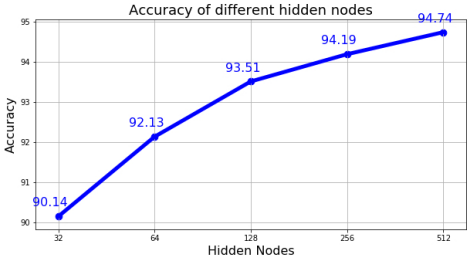
In [25]:
```
plt.plot(train_history_32.history['acc'], linewidth=3)
plt.plot(train_history_64.history['acc'], linewidth=3)
plt.plot(train_history_128.history['acc'], linewidth=3)
plt.plot(train_history_256.history['acc'], linewidth=3)
plt.plot(train_history_512.history['acc'], linewidth=3)
plt.title('Train 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(['32 hidden nodes','64 hidden nodes','128 hidden nodes','256 hidden nodes','512 hidden nodes'], loc='best')
plt.grid(True)
plt.savefig('hidden_nodes_acc_all.jpg',dpi=300)
```



In [26]:
```
plt.plot(train_history_32.history['loss'], linewidth=3)
plt.plot(train_history_64.history['loss'], linewidth=3)
plt.plot(train_history_128.history['loss'], linewidth=3)
plt.plot(train_history_256.history['loss'], linewidth=3)
plt.plot(train_history_512.history['loss'], linewidth=3)
plt.title('Train History')
plt.xlabel('Epoch')
plt.ylabel('Loss')
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(['32 hidden nodes','64 hidden nodes','128 hidden nodes','256 hidden nodes','512 hidden nodes'], loc='best')
plt.grid(True)
plt.savefig('hidden_nodes_loss_all.jpg',dpi=300)
```



In [24]:
```
acc = [round(scores_32[1]*100,2),round(scores_64[1]*100,2),round(scores_128[1]*100,2),round(scores_256[1]*100,2),round(scores
x = ['32','64','128','256','512']
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 nodes', fontsize='18')
plt.xlabel('Hidden Nodes',fontsize=16)
plt.ylabel('Accuracy',fontsize=16)
plt.grid(True)
for x,y in enumerate(acc):
    plt.text(x+0.1,y+0.25,'%s' %y, ha='right', color='b',fontsize=16)
plt.savefig('Accuracy of different hidden nodes.jpg',dpi=300)
```



In [ ]:
```
```