In [1]:
```
from keras.datasets import mnist
from keras import models
from keras import layers
import matplotlib.pyplot as plt
%matplotlib inline
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()
```

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

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]:
```
def show_train_history(train_history, train, validation, size):
    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.xticks([0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19], ['1','2'
    plt.legend(['Train', 'Validation'], loc='best')
    plt.grid(True)
    if train == 'acc':
        plt.savefig("batch_size_acc_" + str(size) + ".jpg")
    if train == 'loss':
        plt.savefig("batch_size_loss_" + str(size) + ".jpg")
    plt.show()
```

In [4]:
```
#batch_size=64
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=['a
train_history_64 = network.fit(train_images, train_labels, validation_split=0
test_loss, test_acc = network.evaluate(test_images, test_labels)
print('test_loss:', test_loss)
print('test_acc:', test_acc)
```

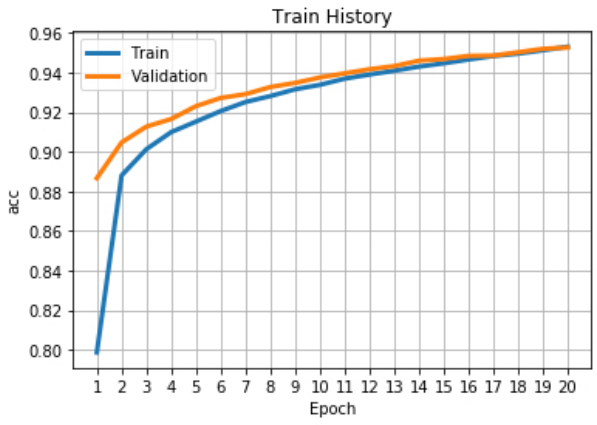
```
WARNING:tensorflow:From D:\Anaconda3\lib\site-packages\tensorflow\python\framew
ork\op_def_library.py:263: colocate_with (from tensorflow.python.framework.ops)
is deprecated and will be removed in a future version.
Instructions for updating:
Colocations handled automatically by placer.
WARNING:tensorflow:From D:\Anaconda3\lib\site-packages\tensorflow\python\ops\ma
th_ops.py:3066: to_int32 (from tensorflow.python.ops.math_ops) is deprecated an
d 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 101us/step - loss: 0.9069 - a
cc: 0.7986 - val_loss: 0.4689 - val_acc: 0.8867
Epoch 2/20
48000/48000 [==============================] - 3s 64us/step - loss: 0.4367 - ac
c: 0.8881 - val_loss: 0.3633 - val_acc: 0.9048
Epoch 3/20
48000/48000 [==============================] - 3s 64us/step - loss: 0.3646 - ac
c: 0.9013 - val_loss: 0.3217 - val_acc: 0.9127
```

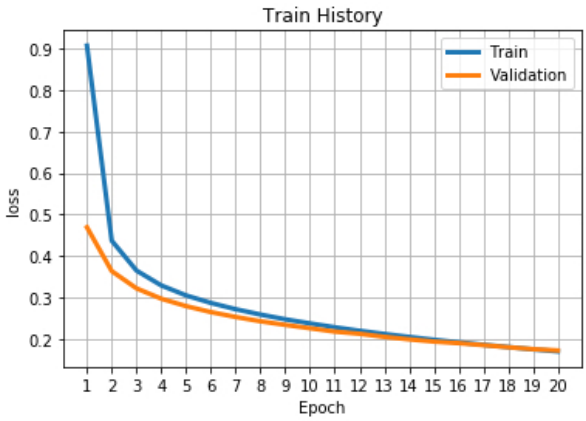
```
Epoch 4/20
48000/48000 [==============================] - 3s 62us/step - loss: 0.3288 - ac
c: 0.9100 - val_loss: 0.2967 - val_acc: 0.9166
Epoch 5/20
48000/48000 [==============================] - 3s 65us/step - loss: 0.3047 - ac
c: 0.9153 - val_loss: 0.2789 - val_acc: 0.9231
Epoch 6/20
48000/48000 [==============================] - 3s 63us/step - loss: 0.2866 - ac
c: 0.9206 - val_loss: 0.2645 - val_acc: 0.9273
Epoch 7/20
48000/48000 [==============================] - 3s 63us/step - loss: 0.2714 - ac
c: 0.9252 - val_loss: 0.2527 - val_acc: 0.9292
Epoch 8/20
48000/48000 [==============================] - 3s 68us/step - loss: 0.2586 - ac
c: 0.9282 - val_loss: 0.2423 - val_acc: 0.9327
Epoch 9/20
48000/48000 [==============================] - 3s 61us/step - loss: 0.2471 - ac
c: 0.9316 - val_loss: 0.2336 - val_acc: 0.9348
Epoch 10/20
48000/48000 [==============================] - 3s 62us/step - loss: 0.2372 - ac
c: 0.9339 - val_loss: 0.2255 - val_acc: 0.9376
Epoch 11/20
48000/48000 [==============================] - 3s 66us/step - loss: 0.2279 - ac
c: 0.9370 - val_loss: 0.2174 - val_acc: 0.9397
Epoch 12/20
48000/48000 [==============================] - 3s 68us/step - loss: 0.2194 - ac
c: 0.9391 - val_loss: 0.2122 - val_acc: 0.9417
Epoch 13/20
48000/48000 [==============================] - 3s 66us/step - loss: 0.2118 - ac
c: 0.9410 - val_loss: 0.2048 - val_acc: 0.9433
Epoch 14/20
48000/48000 [==============================] - 3s 68us/step - loss: 0.2046 - ac
c: 0.9431 - val_loss: 0.1988 - val_acc: 0.9461
Epoch 15/20
48000/48000 [==============================] - 3s 73us/step - loss: 0.1978 - ac
c: 0.9447 - val_loss: 0.1934 - val_acc: 0.9469
Epoch 16/20
48000/48000 [==============================] - 4s 76us/step - loss: 0.1916 - ac
c: 0.9467 - val_loss: 0.1896 - val_acc: 0.9485
Epoch 17/20
48000/48000 [==============================] - 4s 80us/step - loss: 0.1856 - ac
c: 0.9484 - val_loss: 0.1848 - val_acc: 0.9487
Epoch 18/20
48000/48000 [==============================] - 4s 81us/step - loss: 0.1801 - ac
c: 0.9496 - val_loss: 0.1798 - val_acc: 0.9503
Epoch 19/20
48000/48000 [==============================] - 4s 75us/step - loss: 0.1747 - ac
c: 0.9513 - val_loss: 0.1751 - val_acc: 0.9521
Epoch 20/20
48000/48000 [==============================] - 3s 72us/step - loss: 0.1697 - ac
c: 0.9531 - val_loss: 0.1720 - val_acc: 0.9527
10000/10000 [==============================] - 0s 46us/step
test_loss: 0.17141485458612443
test_acc: 0.9505
```

In [5]:

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

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 50us/step

[Info] Accuracy of testing data = 95.3%
```

In [6]:

```
#batch_size=128
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=['a
train_history_128 = network.fit(train_images, train_labels, validation_split=
test_loss, test_acc = network.evaluate(test_images, test_labels)
print('test_loss:', test_loss)
print('test_acc:', test_acc)
```

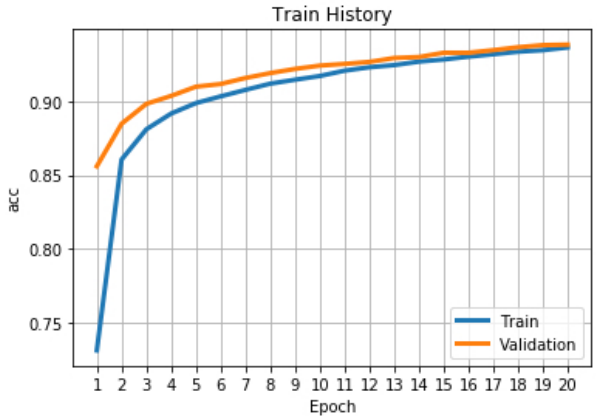
```
Train on 48000 samples, validate on 12000 samples
Epoch 1/20
48000/48000 [==============================] - 2s 45us/step - loss: 1.2046 - ac
c: 0.7309 - val_loss: 0.6742 - val_acc: 0.8560
Epoch 2/20
48000/48000 [==============================] - 2s 39us/step - loss: 0.5887 - ac
c: 0.8606 - val_loss: 0.4705 - val_acc: 0.8849
Epoch 3/20
48000/48000 [==============================] - 2s 38us/step - loss: 0.4642 - ac
c: 0.8811 - val_loss: 0.3991 - val_acc: 0.8984
Epoch 4/20
48000/48000 [==============================] - 2s 38us/step - loss: 0.4081 - ac
c: 0.8918 - val_loss: 0.3617 - val_acc: 0.9038
Epoch 5/20
48000/48000 [==============================] - 2s 38us/step - loss: 0.3746 - ac
c: 0.8989 - val_loss: 0.3370 - val_acc: 0.9101
Epoch 6/20
48000/48000 [==============================] - 2s 39us/step - loss: 0.3511 - ac
c: 0.9035 - val_loss: 0.3190 - val_acc: 0.9119
Epoch 7/20
48000/48000 [==============================] - 2s 38us/step - loss: 0.3334 - ac
c: 0.9079 - val_loss: 0.3055 - val_acc: 0.9160
Epoch 8/20
48000/48000 [==============================] - 2s 37us/step - loss: 0.3188 - ac
c: 0.9121 - val_loss: 0.2937 - val_acc: 0.9193
Epoch 9/20
48000/48000 [==============================] - 2s 39us/step - loss: 0.3068 - ac
c: 0.9148 - val_loss: 0.2842 - val_acc: 0.9222
Epoch 10/20
48000/48000 [==============================] - 2s 40us/step - loss: 0.2964 - ac
c: 0.9174 - val_loss: 0.2756 - val_acc: 0.9244
Epoch 11/20
48000/48000 [==============================] - 2s 40us/step - loss: 0.2870 - ac
c: 0.9210 - val_loss: 0.2679 - val_acc: 0.9256
Epoch 12/20
48000/48000 [==============================] - 2s 39us/step - loss: 0.2784 - ac
c: 0.9233 - val_loss: 0.2613 - val_acc: 0.9269
Epoch 13/20
48000/48000 [==============================] - 2s 40us/step - loss: 0.2708 - ac
c: 0.9248 - val_loss: 0.2548 - val_acc: 0.9296
Epoch 14/20
48000/48000 [==============================] - 2s 39us/step - loss: 0.2637 - ac
c: 0.9271 - val_loss: 0.2490 - val_acc: 0.9303
Epoch 15/20
48000/48000 [==============================] - 2s 38us/step - loss: 0.2570 - ac
c: 0.9286 - val_loss: 0.2434 - val_acc: 0.9332
Epoch 16/20
48000/48000 [==============================] - 2s 40us/step - loss: 0.2508 - ac
c: 0.9305 - val_loss: 0.2390 - val_acc: 0.9332
```

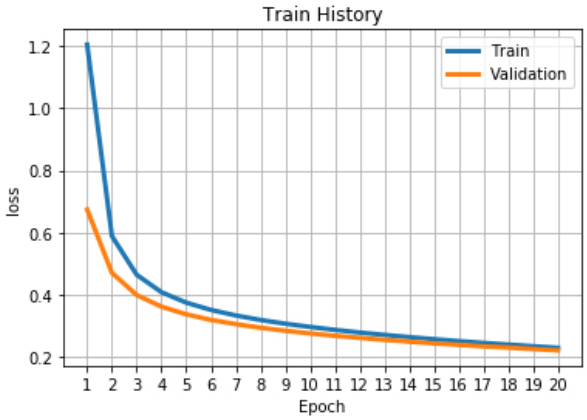
```
Epoch 17/20
48000/48000 [==============================] - 2s 39us/step - loss: 0.2450 - ac
c: 0.9321 - val_loss: 0.2339 - val_acc: 0.9349
Epoch 18/20
48000/48000 [==============================] - 2s 40us/step - loss: 0.2396 - ac
c: 0.9339 - val_loss: 0.2301 - val_acc: 0.9369
Epoch 19/20
48000/48000 [==============================] - 2s 40us/step - loss: 0.2344 - ac
c: 0.9349 - val_loss: 0.2252 - val_acc: 0.9383
Epoch 20/20
48000/48000 [==============================] - 2s 40us/step - loss: 0.2293 - ac
c: 0.9368 - val_loss: 0.2212 - val_acc: 0.9387
10000/10000 [==============================] - 0s 44us/step
test_loss: 0.22186423029005528
test_acc: 0.9392
```

In [7]:

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

scores_128 = network.evaluate(train_images, train_labels)
print()
print("[Info] Accuracy of testing data = {:2.1f}%".format(scores_128[1]*100.
```






```
60000/60000 [==============================] - 2s 41us/step

[Info] Accuracy of testing data = 93.8%
```

In [8]:

```
#batch_size=256
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=['
train_history_256 = network.fit(train_images, train_labels, validation_split=
test_loss, test_acc = network.evaluate(test_images, test_labels)
print('test_loss:', test_loss)
print('test_acc:', test_acc)
```

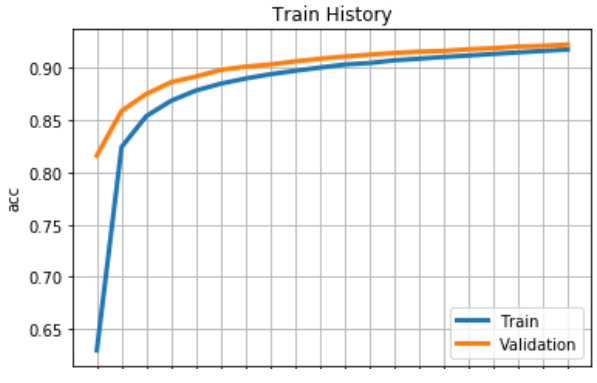
```
Train on 48000 samples, validate on 12000 samples
Epoch 1/20
48000/48000 [==============================] - 1s 27us/step - loss: 1.6042 - ac
c: 0.6296 - val_loss: 1.0446 - val_acc: 0.8163
Epoch 2/20
48000/48000 [==============================] - 1s 22us/step - loss: 0.8617 - ac
c: 0.8244 - val_loss: 0.6791 - val_acc: 0.8591
```

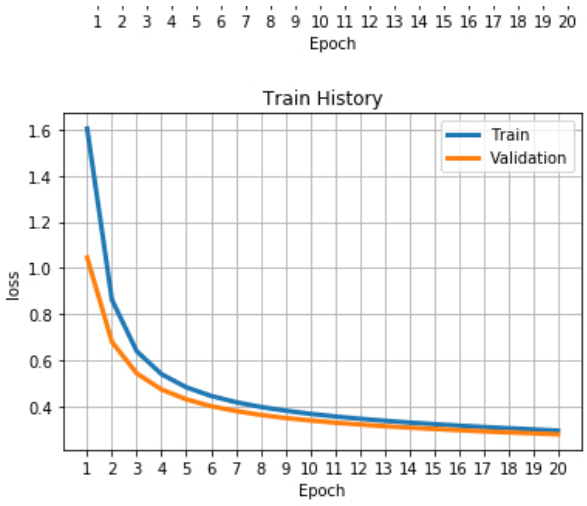
```
Epoch 3/20
48000/48000 [==============================] - 1s 22us/step - loss: 0.6387 - ac
c: 0.8541 - val_loss: 0.5425 - val_acc: 0.8752
Epoch 4/20
48000/48000 [==============================] - 1s 22us/step - loss: 0.5393 - ac
c: 0.8688 - val_loss: 0.4728 - val_acc: 0.8866
Epoch 5/20
48000/48000 [==============================] - 1s 22us/step - loss: 0.4823 - ac
c: 0.8786 - val_loss: 0.4301 - val_acc: 0.8920
Epoch 6/20
48000/48000 [==============================] - 1s 24us/step - loss: 0.4448 - ac
c: 0.8851 - val_loss: 0.4003 - val_acc: 0.8983
Epoch 7/20
48000/48000 [==============================] - 1s 22us/step - loss: 0.4178 - ac
c: 0.8901 - val_loss: 0.3794 - val_acc: 0.9014
Epoch 8/20
48000/48000 [==============================] - 1s 23us/step - loss: 0.3971 - ac
c: 0.8941 - val_loss: 0.3626 - val_acc: 0.9035
Epoch 9/20
48000/48000 [==============================] - 1s 23us/step - loss: 0.3809 - ac
c: 0.8974 - val_loss: 0.3492 - val_acc: 0.9065
Epoch 10/20
48000/48000 [==============================] - 1s 22us/step - loss: 0.3673 - ac
c: 0.9005 - val_loss: 0.3386 - val_acc: 0.9090
Epoch 11/20
48000/48000 [==============================] - 1s 22us/step - loss: 0.3559 - ac
c: 0.9034 - val_loss: 0.3290 - val_acc: 0.9112
Epoch 12/20
48000/48000 [==============================] - 1s 24us/step - loss: 0.3459 - ac
c: 0.9049 - val_loss: 0.3210 - val_acc: 0.9129
Epoch 13/20
48000/48000 [==============================] - 1s 23us/step - loss: 0.3372 - ac
c: 0.9075 - val_loss: 0.3137 - val_acc: 0.9146
Epoch 14/20
48000/48000 [==============================] - 1s 22us/step - loss: 0.3294 - ac
c: 0.9090 - val_loss: 0.3072 - val_acc: 0.9158
Epoch 15/20
48000/48000 [==============================] - 1s 23us/step - loss: 0.3224 - ac
c: 0.9107 - val_loss: 0.3019 - val_acc: 0.9165
Epoch 16/20
48000/48000 [==============================] - 1s 22us/step - loss: 0.3160 - ac
c: 0.9121 - val_loss: 0.2962 - val_acc: 0.9181
Epoch 17/20
48000/48000 [==============================] - 1s 23us/step - loss: 0.3101 - ac
c: 0.9136 - val_loss: 0.2912 - val_acc: 0.9191
Epoch 18/20
48000/48000 [==============================] - 1s 22us/step - loss: 0.3046 - ac
c: 0.9151 - val_loss: 0.2867 - val_acc: 0.9208
Epoch 19/20
48000/48000 [==============================] - 1s 22us/step - loss: 0.2995 - ac
c: 0.9165 - val_loss: 0.2825 - val_acc: 0.9214
Epoch 20/20
48000/48000 [==============================] - 1s 22us/step - loss: 0.2946 - ac
c: 0.9179 - val_loss: 0.2786 - val_acc: 0.9226
10000/10000 [==============================] - 0s 46us/step
test_loss: 0.27871255747675894
test_acc: 0.9238
```

In [9]:

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

scores_256 = network.evaluate(train_images, train_labels)
print()
print("[Info] Accuracy of testing data = {:2.1f}%".format(scores_256[1]*100.
```






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

[Info] Accuracy of testing data = 92.0%
```

In [10]:

```python
#batch_size=512
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=['a
train_history_512 = network.fit(train_images, train_labels, validation_split=
test_loss, test_acc = network.evaluate(test_images, test_labels)
print('test_loss:', test_loss)
print('test_acc:', test_acc)
```

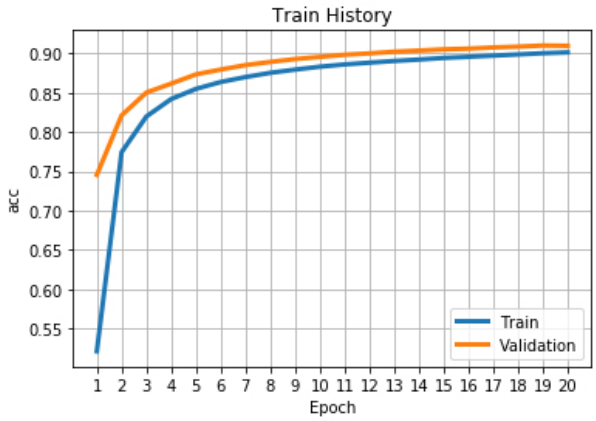
```
Train on 48000 samples, validate on 12000 samples
Epoch 1/20
48000/48000 [==============================] - 1s 23us/step - loss: 1.8946 - ac
c: 0.5212 - val_loss: 1.5104 - val_acc: 0.7456
Epoch 2/20
48000/48000 [==============================] - 1s 17us/step - loss: 1.2810 - ac
c: 0.7745 - val_loss: 1.0482 - val_acc: 0.8213
Epoch 3/20
48000/48000 [==============================] - 1s 17us/step - loss: 0.9487 - ac
c: 0.8199 - val_loss: 0.8087 - val_acc: 0.8504
Epoch 4/20
48000/48000 [==============================] - 1s 17us/step - loss: 0.7721 - ac
c: 0.8421 - val_loss: 0.6760 - val_acc: 0.8617
Epoch 5/20
48000/48000 [==============================] - 1s 17us/step - loss: 0.6680 - ac
c: 0.8550 - val_loss: 0.5936 - val_acc: 0.8735
Epoch 6/20
48000/48000 [==============================] - 1s 17us/step - loss: 0.6003 - ac
c: 0.8638 - val_loss: 0.5385 - val_acc: 0.8797
Epoch 7/20
48000/48000 [==============================] - 1s 17us/step - loss: 0.5526 - ac
c: 0.8701 - val_loss: 0.4991 - val_acc: 0.8853
Epoch 8/20
48000/48000 [==============================] - 1s 17us/step - loss: 0.5171 - ac
c: 0.8755 - val_loss: 0.4692 - val_acc: 0.8893
Epoch 9/20
48000/48000 [==============================] - 1s 18us/step - loss: 0.4896 - ac
c: 0.8796 - val_loss: 0.4458 - val_acc: 0.8930
Epoch 10/20
48000/48000 [==============================] - 1s 17us/step - loss: 0.4676 - ac
c: 0.8833 - val_loss: 0.4271 - val_acc: 0.8958
Epoch 11/20
48000/48000 [==============================] - 1s 17us/step - loss: 0.4494 - ac
c: 0.8862 - val_loss: 0.4115 - val_acc: 0.8985
Epoch 12/20
48000/48000 [==============================] - 1s 17us/step - loss: 0.4342 - ac
c: 0.8884 - val_loss: 0.3985 - val_acc: 0.9003
Epoch 13/20
48000/48000 [==============================] - 1s 17us/step - loss: 0.4211 - ac
c: 0.8906 - val_loss: 0.3874 - val_acc: 0.9023
Epoch 14/20
48000/48000 [==============================] - 1s 19us/step - loss: 0.4098 - ac
c: 0.8924 - val_loss: 0.3778 - val_acc: 0.9038
Epoch 15/20
48000/48000 [==============================] - 1s 18us/step - loss: 0.3997 - ac
c: 0.8944 - val_loss: 0.3691 - val_acc: 0.9054
```

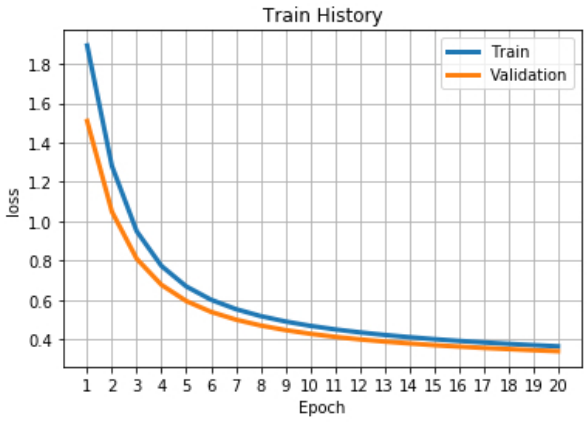
```
Epoch 16/20
48000/48000 [==============================] - 1s 18us/step - loss: 0.3908 - ac
c: 0.8959 - val_loss: 0.3619 - val_acc: 0.9062
Epoch 17/20
48000/48000 [==============================] - 1s 17us/step - loss: 0.3830 - ac
c: 0.8974 - val_loss: 0.3549 - val_acc: 0.9078
Epoch 18/20
48000/48000 [==============================] - 1s 17us/step - loss: 0.3758 - ac
c: 0.8989 - val_loss: 0.3491 - val_acc: 0.9088
Epoch 19/20
48000/48000 [==============================] - 1s 17us/step - loss: 0.3692 - ac
c: 0.9004 - val_loss: 0.3432 - val_acc: 0.9102
Epoch 20/20
48000/48000 [==============================] - 1s 17us/step - loss: 0.3632 - ac
c: 0.9015 - val_loss: 0.3382 - val_acc: 0.9099
10000/10000 [==============================] - 0s 49us/step
test_loss: 0.3410462572097778
test_acc: 0.9079
```

In [11]:

```
size = 512
show_train_history(train_history_512, 'acc', 'val_acc', size)
show_train_history(train_history_512, 'loss', 'val_loss', size)

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 [==============================] - 3s 47us/step

[Info] Accuracy of testing data = 90.4%
```

In [12]:

```
#batch_size=1024
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=['a
train_history_1024 = network.fit(train_images, train_labels, validation_spli
test_loss, test_acc = network.evaluate(test_images, test_labels)
print('test_loss:', test_loss)
print('test_acc:', test_acc)
```

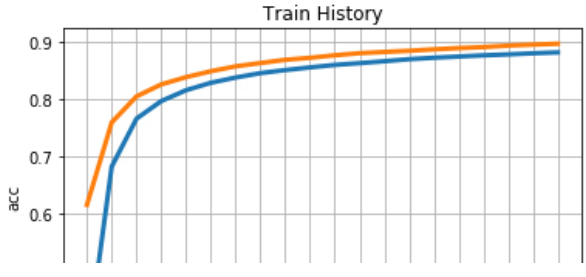
```
Train on 48000 samples, validate on 12000 samples
Epoch 1/20
48000/48000 [==============================] - 1s 20us/step - loss: 2.0580 - ac
c: 0.3625 - val_loss: 1.8132 - val_acc: 0.6155
```

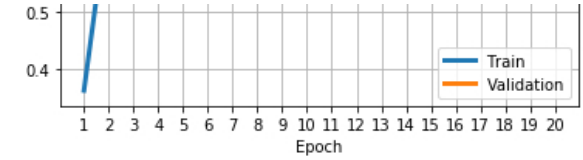
```
Epoch 2/20
48000/48000 [==============================] - 1s 13us/step - loss: 1.6529 - ac
c: 0.6825 - val_loss: 1.4608 - val_acc: 0.7597
Epoch 3/20
48000/48000 [==============================] - 1s 13us/step - loss: 1.3537 - ac
c: 0.7663 - val_loss: 1.2007 - val_acc: 0.8052
Epoch 4/20
48000/48000 [==============================] - 1s 13us/step - loss: 1.1373 - ac
c: 0.7970 - val_loss: 1.0154 - val_acc: 0.8262
Epoch 5/20
48000/48000 [==============================] - 1s 13us/step - loss: 0.9833 - ac
c: 0.8160 - val_loss: 0.8837 - val_acc: 0.8388
Epoch 6/20
48000/48000 [==============================] - 1s 13us/step - loss: 0.8725 - ac
c: 0.8290 - val_loss: 0.7877 - val_acc: 0.8492
Epoch 7/20
48000/48000 [==============================] - 1s 13us/step - loss: 0.7906 - ac
c: 0.8382 - val_loss: 0.7163 - val_acc: 0.8578
Epoch 8/20
48000/48000 [==============================] - 1s 13us/step - loss: 0.7282 - ac
c: 0.8459 - val_loss: 0.6615 - val_acc: 0.8635
Epoch 9/20
48000/48000 [==============================] - 1s 13us/step - loss: 0.6793 - ac
c: 0.8511 - val_loss: 0.6182 - val_acc: 0.8692
Epoch 10/20
48000/48000 [==============================] - 1s 13us/step - loss: 0.6400 - ac
c: 0.8559 - val_loss: 0.5832 - val_acc: 0.8727
Epoch 11/20
48000/48000 [==============================] - 1s 14us/step - loss: 0.6077 - ac
c: 0.8602 - val_loss: 0.5543 - val_acc: 0.8771
Epoch 12/20
48000/48000 [==============================] - 1s 15us/step - loss: 0.5808 - ac
c: 0.8633 - val_loss: 0.5303 - val_acc: 0.8808
Epoch 13/20
48000/48000 [==============================] - 1s 14us/step - loss: 0.5579 - ac
c: 0.8666 - val_loss: 0.5097 - val_acc: 0.8829
Epoch 14/20
48000/48000 [==============================] - 1s 14us/step - loss: 0.5382 - ac
c: 0.8702 - val_loss: 0.4921 - val_acc: 0.8849
Epoch 15/20
48000/48000 [==============================] - 1s 13us/step - loss: 0.5210 - ac
c: 0.8728 - val_loss: 0.4767 - val_acc: 0.8875
Epoch 16/20
48000/48000 [==============================] - 1s 14us/step - loss: 0.5059 - ac
c: 0.8749 - val_loss: 0.4633 - val_acc: 0.8897
Epoch 17/20
48000/48000 [==============================] - 1s 13us/step - loss: 0.4925 - ac
c: 0.8770 - val_loss: 0.4514 - val_acc: 0.8914
Epoch 18/20
48000/48000 [==============================] - 1s 13us/step - loss: 0.4806 - ac
c: 0.8786 - val_loss: 0.4406 - val_acc: 0.8940
Epoch 19/20
48000/48000 [==============================] - 1s 13us/step - loss: 0.4698 - ac
c: 0.8805 - val_loss: 0.4310 - val_acc: 0.8958
Epoch 20/20
48000/48000 [==============================] - 1s 13us/step - loss: 0.4600 - ac
c: 0.8821 - val_loss: 0.4224 - val_acc: 0.8973
10000/10000 [==============================] - 0s 46us/step
test_loss: 0.4272878079652786
test_acc: 0.8952
```

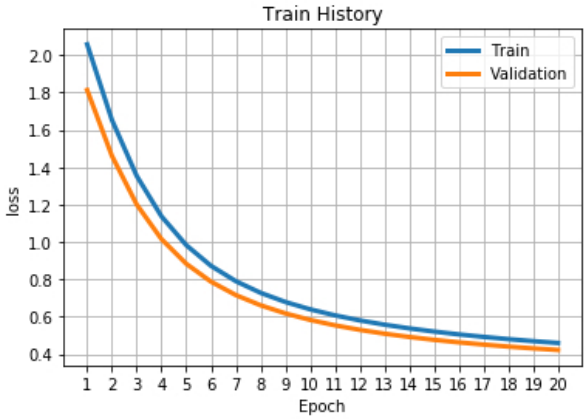
In [13]:

```
size = 1024
show_train_history(train_history_1024, 'acc', 'val_acc', size)
show_train_history(train_history_1024, 'loss', 'val_loss', size)


scores_1024 = network.evaluate(train_images, train_labels)
print()
print("[Info] Accuracy of testing data = {:2.1f}%".format(scores_1024[1]*100
```





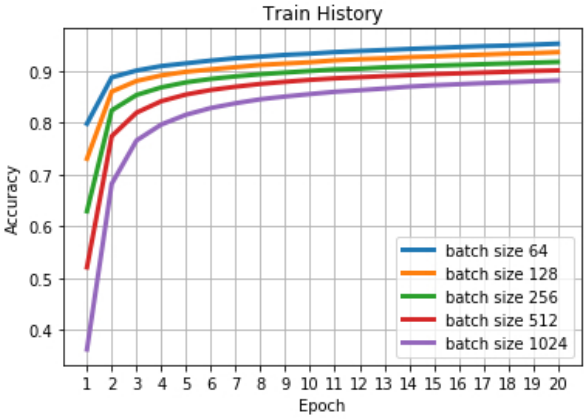



```
60000/60000 [==============================] - 3s 45us/step

[Info] Accuracy of testing data = 88.6%
```

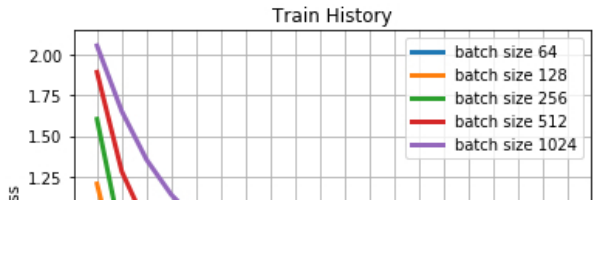
In [14]:

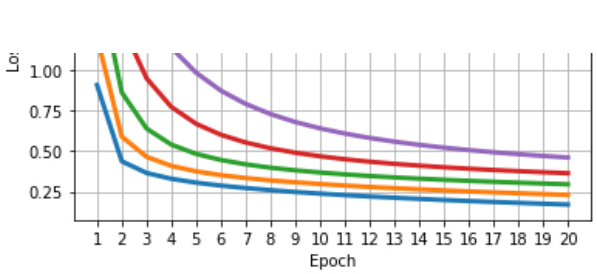
```
plt.plot(train_history_64.history['acc'], linewidth=3)
plt.plot(train_history_128.history['acc'], linewidth=3)
plt.plot(train_history_256.history['acc'], linewidth=3)
plt.plot(train_history_512.history['acc'], linewidth=3)
plt.plot(train_history_1024.history['acc'], linewidth=3)
plt.title('Train 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',
plt.legend(['batch size 64','batch size 128','batch size 256','batch size 512
plt.grid(True)
plt.savefig('batch_size_acc_all.jpg',dpi=300)
```



In [15]:

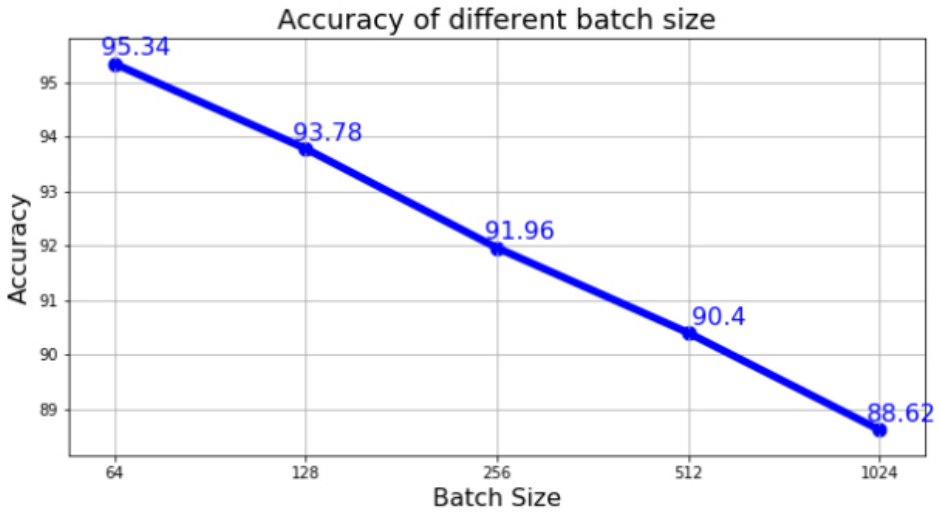
```
plt.plot(train_history_64.history['loss'], linewidth=3)
plt.plot(train_history_128.history['loss'], linewidth=3)
plt.plot(train_history_256.history['loss'], linewidth=3)
plt.plot(train_history_512.history['loss'], linewidth=3)
plt.plot(train_history_1024.history['loss'], linewidth=3)
plt.title('Train 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',
plt.legend(['batch size 64','batch size 128','batch size 256','batch size 512
plt.grid(True)
plt.savefig('batch_size_loss_all.jpg',dpi=300)
```





In [53]:

```
acc = [round(scores_64[1]*100,2),round(scores_128[1]*100,2),round(scores_256
x = ['64','128','256','512','1024']
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 batch size', fontsize='18')
plt.xlabel('Batch Size',fontsize=16)
plt.ylabel('Accuracy',fontsize=16)
plt.grid(True)
for x,y in enumerate(acc):
    plt.text(x+0.3,y+0.15,'%s' %y, ha='right', color='b',fontsize=16)
plt.savefig('Accuracy of different batch size.jpg',dpi=300)
```



In [ ]:

```
1
```