hidden act_func	L2_norm	dropout	accuracy
ReLU	true	true	76.86
ReLU	true	false	78.33
ReLU	false	true	77.38
ReLU	false	false	75.22
sigmoid	true	true	74.14
sigmoid	true	false	50.00
sigmoid	false	true	79.44
sigmoid	false	false	75.52
tanh	true	true	74.62
tanh	true	false	77.58
tanh	false	true	79.75
tanh	false	false	76.59

- **What effect do activation functions have on your results?**

  These three activation functions perform similarly, while ***tanh*** seems the best for this data.

  I think that *tanh* is suitable for classifying problem because its function is $tanh(x) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$, which means approximating a classifier function as combinations of tanh is easier than *ReLu*.

  And actually, *sigmoid* also works well for a classifier but it has gradient vanishing and zero-centered problems.

  For *ReLU*, it leads to faster training process and convergence without gradient vanishing, which is shown in my results below.

- **What effect does addition of *L2-norm* regularization have on the results?**

  With *L2-norm*, the results aren't improved, and even **worse** sometimes. Since *L2-norm* penalties in some sense discourage sparsity by yielding diminishing returns as elements are moved closer to 0, while our data is sparse. This worsen *sigmoid* 's zero-centered problem.

- **What effect does dropout have on the results?**

  With *dropout*, the model performs **better**. *dropout* prevent over-fitting, reducing interdependent learning amongst the neurons. It helps three action functions, especially *sigmoid*, perform better.

In [1]:
```
import sys
import numpy as np
import pandas as pd

from keras import Sequential
from keras.layers import Input, Dense, LSTM, Embedding, Dropout, BatchNormalization, Activation, Bidirectional,Flatten
from keras import optimizers
from keras import regularizers
from keras.utils import np_utils
from keras.preprocessing.text import text_to_word_sequence, Tokenizer
from keras.preprocessing.sequence import pad_sequences

from gensim.models import Word2Vec
from sklearn.model_selection import train_test_split

DATA_DIR = "./"
```

```
C:\Users\h5weng\AppData\Local\Continuum\anaconda3\lib\site-packages\h5py\__init__.py:34: 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.
C:\Users\h5weng\AppData\Local\Continuum\anaconda3\lib\site-packages\gensim\utils.py:1197: UserWarning: detected Windows; aliasing chunkize to chunkize_serial
  warnings.warn("detected Windows; aliasing chunkize to chunkize_serial")
```

In [2]:
```
# load data from files and save it into lists
def load_data(f, data, pos_or_neg, target):
    for line in f.readlines():
        line_str = ""
        line = line[1:-2].split(r", ")
        for i in range(len(line)):
            line[i] = line[i][1:-1]
            line_str += str(line[i]+" ")
        data.append(line_str)
        target.append(pos_or_neg)

data = []
target = []

for file_name in ["training_pos.csv",
          "training_neg.csv",
          "validation_pos.csv",
          "validation_neg.csv",
          "test_pos.csv",
          "test_neg.csv"]:
    f = open(DATA_DIR + file_name,'r')
    pos_or_neg = 1 if 'pos' in file_name else 0
    load_data(f,data, pos_or_neg, target)
    f.close()
```

In [3]:
```
word_seq = [text_to_word_sequence(sent) for sent in data]
# we should be safe using MAX _SENT _LEN as 90th Percentile Sentence Length
MAX_SENT_LEN = int(np.percentile([len(seq) for seq in word_seq], 90))
print('90th Percentile Sentence Length:', MAX_SENT_LEN)

# cut every sentence  according to MAX_SENT_LEN
data = [' '.join(seq[:MAX_SENT_LEN]) for seq in word_seq]

# Convert the sequence of words to sequnce of indices
MAX_VOCAB_SIZE = 80000
tokenizer = Tokenizer(MAX_VOCAB_SIZE)
tokenizer.fit_on_texts(data)
data = tokenizer.texts_to_sequences(data)
data = pad_sequences(data, maxlen=MAX_SENT_LEN, padding='post', truncating='post')
```

```
90th Percentile Sentence Length: 23
```

In [5]:
```
# split data into train, validation, test
length = len(data)
train_data = data[:int(length*0.8)]
val_data = data[int(length*0.9):]
test_data = data[int(length*0.8):int(length*0.9)]
train_target = target[:int(length*0.8)]
val_target = target[int(length*0.9):]
test_target = target[int(length*0.8):int(length*0.9)]

train = list(zip(train_data, train_target))
import random
```

```
import random
random.shuffle(train)
train_data[:], train_target[:] = zip(*train)
```

In [6]:
```
train_target = np_utils.to_categorical(train_target, 2)
test_target =  np_utils.to_categorical(test_target, 2)
val_target = np_utils.to_categorical(val_target, 2)
```

In [7]:
```
W2V_DIR = 'W2V_DIR'
embeddings = Word2Vec.load(DATA_DIR + W2V_DIR)
print('Dimension of w2v:', embeddings.vector_size)
EMBEDDING_DIM = embeddings.vector_size
```

```
C:\Users\h5weng\AppData\Local\Continuum\anaconda3\lib\site-packages\smart_open\smart_open_lib.py:398: UserWarning: This function
is deprecated, use smart_open.open instead. See the migration notes for details: https://github.com/RaRe-Technologies/smart_ope
n/blob/master/README.rst#migrating-to-the-new-open-function
  'See the migration notes for details: %s' % _MIGRATION_NOTES_URL
```

```
Dimension of w2v: 300
```

In [8]:
```
# Create an embedding matrix containing only the word's in our vocabulary
# If the word does not have a pre-trained embedding, then randomly initialize the embedding
embeddings_matrix = np.random.uniform(-0.05, 0.05, size=(len(tokenizer.word_index)+1, EMBEDDING_DIM)) # +1 is because the matrix
indices start with 0

for word, i in tokenizer.word_index.items(): # i=0 is the embedding for the zero padding
    try:
        embeddings_vector = embeddings[word]
    except KeyError:
        embeddings_vector = None
    if embeddings_vector is not None:
        embeddings_matrix[i] = embeddings_vector

del embeddings
```

```
C:\Users\h5weng\AppData\Local\Continuum\anaconda3\lib\site-packages\ipykernel_launcher.py:7: DeprecationWarning: Call to depreca
ted `__getitem__` (Method will be removed in 4.0.0, use self.wv.__getitem__() instead).
  import sys
```

In [9]:
```
def sequential_model (hidden_actfunc, l2normFlag, dropoutFlag, dropoutRate):

    # Build a sequential model by stacking neural net units
    model = Sequential()
    # Input layer of embeddings
    model.add(Embedding(input_dim=len(tokenizer.word_index)+1,
                            output_dim=EMBEDDING_DIM,
                            weights = [embeddings_matrix], trainable=True, name='word_embedding_layer',
                            mask_zero=False,input_length=MAX_SENT_LEN))
#     model.add(LSTM(128, return_sequences=False, name='lstm_layer'))
    model.add(Flatten())

    if(dropoutFlag):
        model.add(Dropout(dropoutRate))

    if(l2normFlag):
        model.add(Dense(units = 30,activation = hidden_actfunc, name = 'hidden_layer',kernel_regularizer=regularizers.l2(l=0.1
)))
    else:
        model.add(Dense(units = 30,activation = hidden_actfunc, name = 'hidden_layer'))

    if(dropoutFlag):
        model.add(Dropout(dropoutRate))

    if(l2normFlag):
        model.add(Dense(2, activation='softmax', name='output_layer',kernel_regularizer=regularizers.l2(l=0.1)))
    else:
        model.add(Dense(2, activation='softmax', name='output_layer'))

    # Use cross-entropy as the loss function
    model.compile(loss='categorical_crossentropy',optimizer ='adam',  metrics=['accuracy'])
    return model
```

In [10]:
```
# from keras.utils import plot_model
# from IPython.display import Image
# plot_model(model, to_file='basic_lstm_classifier.png', show_layer_names=True, show_shapes=True)
# Image('basic_lstm_classifier.png')
```

In [11]:
```
# # train data
# BATCH_SIZE = 128
# N_EPOCHS = 10
# dropoutRate = 0.5
# for hidden_actfunc in ["relu", "sigmoid", "tanh"]:
#     for l2normFlag in [True, False]:
#         for dropoutFlag in [True, False]:
#             print("-------------------------------------------------------------------------------")
#             print("hidden_actfunc: "+hidden_actfunc, "  l2normFlag: " + str(l2normFlag),
#                       "  dropoutFlag: "+str(dropoutFlag), "  dropoutRate " + str(dropoutRate))
#             for dropoutRate in np.arange(0.4, 1 , 0.2):
#                 model = sequential_model (hidden_actfunc, l2normFlag, dropoutFlag, dropoutRate)
#                 model.fit(train_data, train_target,batch_size=BATCH_SIZE,epochs=N_EPOCHS, validation_data =(val_data, val_targ
```

```
et))
```

In [12]:
```
BATCH_SIZE = 128
N_EPOCHS = 10
```

relu

In [13]:
```
# hidden_layer = relu, l2norm = True, dropout = True
hidden_actfunc = "relu"
l2normFlag = True
dropoutFlag = True
dropoutRate = 0.5
model = sequential_model (hidden_actfunc, l2normFlag, dropoutFlag, dropoutRate)
print("--------------------------------------------------------------------------------")
print("hidden_actfunc: "+hidden_actfunc, "  l2normFlag: " + str(l2normFlag),
      "  dropoutFlag: "+str(dropoutFlag), "  dropoutRate " + str(dropoutRate))
model.fit(train_data, train_target,batch_size=BATCH_SIZE,epochs=N_EPOCHS, validation_data =(val_data, val_target))
scores = model.evaluate(test_data, test_target)
print("%s: %.2f%%"%(model.metrics_names[1], scores[1]*100))
```

```
WARNING: Logging before flag parsing goes to stderr.
W0625 12:20:14.654095 113336 deprecation_wrapper.py:119] From C:\Users\h5weng\AppData\Local\Continuum\anaconda3\lib\site-package
s\keras\backend\tensorflow_backend.py:74: The name tf.get_default_graph is deprecated. Please use tf.compat.v1.get_default_graph
instead.

W0625 12:20:14.682020 113336 deprecation_wrapper.py:119] From C:\Users\h5weng\AppData\Local\Continuum\anaconda3\lib\site-package
s\keras\backend\tensorflow_backend.py:517: The name tf.placeholder is deprecated. Please use tf.compat.v1.placeholder instead.

W0625 12:20:14.691023 113336 deprecation_wrapper.py:119] From C:\Users\h5weng\AppData\Local\Continuum\anaconda3\lib\site-package
s\keras\backend\tensorflow_backend.py:4138: The name tf.random_uniform is deprecated. Please use tf.random.uniform instead.

W0625 12:20:14.701988 113336 deprecation_wrapper.py:119] From C:\Users\h5weng\AppData\Local\Continuum\anaconda3\lib\site-package
s\keras\backend\tensorflow_backend.py:174: The name tf.get_default_session is deprecated. Please use tf.compat.v1.get_default_se
ssion instead.

W0625 12:20:14.702964 113336 deprecation_wrapper.py:119] From C:\Users\h5weng\AppData\Local\Continuum\anaconda3\lib\site-package
s\keras\backend\tensorflow_backend.py:181: The name tf.ConfigProto is deprecated. Please use tf.compat.v1.ConfigProto instead.

W0625 12:20:17.631944 113336 deprecation.py:506] From C:\Users\h5weng\AppData\Local\Continuum\anaconda3\lib\site-packages\keras
\backend\tensorflow_backend.py:3445: calling dropout (from tensorflow.python.ops.nn_ops) with keep_prob is deprecated and will b
e removed in a future version.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 - keep_prob`.
W0625 12:20:17.679821 113336 deprecation_wrapper.py:119] From C:\Users\h5weng\AppData\Local\Continuum\anaconda3\lib\site-package
s\keras\optimizers.py:790: The name tf.train.Optimizer is deprecated. Please use tf.compat.v1.train.Optimizer instead.

W0625 12:20:17.781545 113336 deprecation.py:323] From C:\Users\h5weng\AppData\Local\Continuum\anaconda3\lib\site-packages\tensor
flow\python\ops\math_grad.py:1250: add_dispatch_support.<locals>.wrapper (from tensorflow.python.ops.array_ops) is deprecated an
d will be removed in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where

--------------------------------------------------------------------------------
hidden_actfunc: relu   l2normFlag: True   dropoutFlag: True   dropoutRate 0.5
Train on 640000 samples, validate on 80000 samples
Epoch 1/10
640000/640000 [==============================] - 131s 204us/step - loss: 0.7795 - acc: 0.5416 - val_loss: 0.7189 - val_acc: 0.66
40
Epoch 2/10
640000/640000 [==============================] - 129s 202us/step - loss: 0.7454 - acc: 0.5826 - val_loss: 0.6916 - val_acc: 0.71
40
Epoch 3/10
640000/640000 [==============================] - 129s 202us/step - loss: 0.7396 - acc: 0.5992 - val_loss: 0.6745 - val_acc: 0.73
33
Epoch 4/10
640000/640000 [==============================] - 130s 203us/step - loss: 0.7342 - acc: 0.6081 - val_loss: 0.6603 - val_acc: 0.75
64
Epoch 5/10
640000/640000 [==============================] - 129s 202us/step - loss: 0.7301 - acc: 0.6130 - val_loss: 0.6594 - val_acc: 0.76
39
Epoch 6/10
640000/640000 [==============================] - 130s 203us/step - loss: 0.7268 - acc: 0.6172 - val_loss: 0.6457 - val_acc: 0.76
68
Epoch 7/10
640000/640000 [==============================] - 130s 203us/step - loss: 0.7242 - acc: 0.6206 - val_loss: 0.6478 - val_acc: 0.77
03
Epoch 8/10
640000/640000 [==============================] - 130s 203us/step - loss: 0.7217 - acc: 0.6232 - val_loss: 0.6491 - val_acc: 0.76
47
Epoch 9/10
640000/640000 [==============================] - 129s 202us/step - loss: 0.7202 - acc: 0.6254 - val_loss: 0.6319 - val_acc: 0.77
45
Epoch 10/10
640000/640000 [==============================] - 129s 202us/step - loss: 0.7182 - acc: 0.6266 - val_loss: 0.6308 - val_acc: 0.77
45
80000/80000 [==============================] - 3s 35us/step
acc: 76.86%
```

In [14]:
```
# hidden_layer = relu, l2norm = True, dropout = False
hidden_actfunc = "relu"
l2normFlag = True
dropoutFlag = False
dropoutRate = 0.5
model = sequential_model (hidden_actfunc, l2normFlag, dropoutFlag, dropoutRate)
print("--------------------------------------------------------------------------------")
print("hidden_actfunc: "+hidden_actfunc, "  l2normFlag: " + str(l2normFlag),
      "  dropoutFlag: "+str(dropoutFlag), "  dropoutRate " + str(dropoutRate))
model.fit(train_data, train_target,batch_size=BATCH_SIZE,epochs=N_EPOCHS, validation_data =(val_data, val_target))
scores = model.evaluate(test_data, test_target)
print("%s: %.2f%%"%(model.metrics_names[1], scores[1]*100))
```

```
--------------------------------------------------------------------------------
hidden_actfunc: relu   l2normFlag: True   dropoutFlag: False   dropoutRate 0.5
Train on 640000 samples, validate on 80000 samples
Epoch 1/10
640000/640000 [==============================] - 128s 201us/step - loss: 0.7305 - acc: 0.5818 - val_loss: 0.6240 - val_acc: 0.72
60
Epoch 2/10
640000/640000 [==============================] - 127s 199us/step - loss: 0.6782 - acc: 0.6204 - val_loss: 0.5905 - val_acc: 0.76
15
Epoch 3/10
640000/640000 [==============================] - 127s 199us/step - loss: 0.6706 - acc: 0.6312 - val_loss: 0.5895 - val_acc: 0.77
11
Epoch 4/10
640000/640000 [==============================] - 127s 199us/step - loss: 0.6658 - acc: 0.6364 - val_loss: 0.5726 - val_acc: 0.77
92
Epoch 5/10
640000/640000 [==============================] - 127s 199us/step - loss: 0.6625 - acc: 0.6400 - val_loss: 0.5650 - val_acc: 0.78
33
Epoch 6/10
640000/640000 [==============================] - 128s 199us/step - loss: 0.6598 - acc: 0.6427 - val_loss: 0.5603 - val_acc: 0.78
02
Epoch 7/10
640000/640000 [==============================] - 120s 187us/step - loss: 0.6576 - acc: 0.6452 - val_loss: 0.5659 - val_acc: 0.78
23
Epoch 8/10
640000/640000 [==============================] - 119s 187us/step - loss: 0.6559 - acc: 0.6474 - val_loss: 0.5620 - val_acc: 0.78
56
Epoch 9/10
640000/640000 [==============================] - 119s 186us/step - loss: 0.6543 - acc: 0.6494 - val_loss: 0.5480 - val_acc: 0.78
40
Epoch 10/10
640000/640000 [==============================] - 119s 186us/step - loss: 0.6531 - acc: 0.6509 - val_loss: 0.5650 - val_acc: 0.78
40
80000/80000 [==============================] - 3s 35us/step
acc: 78.33%
```

In [15]:
```
# hidden_layer = relu, l2norm = False, dropout = True
hidden_actfunc = "relu"
l2normFlag = False
dropoutFlag = True
dropoutRate = 0.5
model = sequential_model (hidden_actfunc, l2normFlag, dropoutFlag, dropoutRate)
print("--------------------------------------------------------------------------------")
print("hidden_actfunc: "+hidden_actfunc, "  l2normFlag: " + str(l2normFlag),
      "  dropoutFlag: "+str(dropoutFlag), "  dropoutRate " + str(dropoutRate))
model.fit(train_data, train_target,batch_size=BATCH_SIZE,epochs=N_EPOCHS, validation_data =(val_data, val_target))
scores = model.evaluate(test_data, test_target)
print("%s: %.2f%%"%(model.metrics_names[1], scores[1]*100))
```

```
--------------------------------------------------------------------------------
hidden_actfunc: relu   l2normFlag: False   dropoutFlag: True   dropoutRate 0.5
Train on 640000 samples, validate on 80000 samples
Epoch 1/10
640000/640000 [==============================] - 123s 192us/step - loss: 0.6717 - acc: 0.5830 - val_loss: 0.5816 - val_acc: 0.75
64
Epoch 2/10
640000/640000 [==============================] - 122s 191us/step - loss: 0.6549 - acc: 0.6155 - val_loss: 0.5626 - val_acc: 0.77
22
Epoch 3/10
640000/640000 [==============================] - 122s 191us/step - loss: 0.6493 - acc: 0.6255 - val_loss: 0.5825 - val_acc: 0.76
39
Epoch 4/10
640000/640000 [==============================] - 122s 191us/step - loss: 0.6452 - acc: 0.6316 - val_loss: 0.5581 - val_acc: 0.78
19
Epoch 5/10
640000/640000 [==============================] - 122s 191us/step - loss: 0.6415 - acc: 0.6361 - val_loss: 0.5496 - val_acc: 0.78
43
Epoch 6/10
640000/640000 [==============================] - 122s 191us/step - loss: 0.6379 - acc: 0.6408 - val_loss: 0.5133 - val_acc: 0.78
91
Epoch 7/10
640000/640000 [==============================] - 122s 191us/step - loss: 0.6341 - acc: 0.6443 - val_loss: 0.5383 - val_acc: 0.78
01
Epoch 8/10
640000/640000 [==============================] - 122s 191us/step - loss: 0.6305 - acc: 0.6480 - val_loss: 0.5270 - val_acc: 0.78
44
Epoch 9/10
640000/640000 [==============================] - 122s 191us/step - loss: 0.6270 - acc: 0.6503 - val_loss: 0.5440 - val_acc: 0.77
27
```

```
Epoch 10/10
640000/640000 [==============================] - 122s 191us/step - loss: 0.6233 - acc: 0.6544 - val_loss: 0.5386 - val_acc: 0.77
54
80000/80000 [==============================] - 3s 35us/step
acc: 77.38%
```

In [16]:
```
# hidden_layer = relu, l2norm = False, dropout = False
hidden_actfunc = "relu"
l2normFlag = False
dropoutFlag = False
dropoutRate = 0.5
model = sequential_model (hidden_actfunc, l2normFlag, dropoutFlag, dropoutRate)
print("--------------------------------------------------------------------------------")
print("hidden_actfunc: "+hidden_actfunc, "  l2normFlag: " + str(l2normFlag),
      "  dropoutFlag: "+str(dropoutFlag), "  dropoutRate " + str(dropoutRate))
model.fit(train_data, train_target,batch_size=BATCH_SIZE,epochs=N_EPOCHS, validation_data =(val_data, val_target))
scores = model.evaluate(test_data, test_target)
print("%s: %.2f%%"%(model.metrics_names[1], scores[1]*100))
```

```
--------------------------------------------------------------------------------
hidden_actfunc: relu   l2normFlag: False   dropoutFlag: False   dropoutRate 0.5
Train on 640000 samples, validate on 80000 samples
Epoch 1/10
640000/640000 [==============================] - 120s 188us/step - loss: 0.6511 - acc: 0.6163 - val_loss: 0.5342 - val_acc: 0.78
34
Epoch 2/10
640000/640000 [==============================] - 119s 187us/step - loss: 0.6301 - acc: 0.6472 - val_loss: 0.5267 - val_acc: 0.79
05
Epoch 3/10
640000/640000 [==============================] - 119s 187us/step - loss: 0.6175 - acc: 0.6607 - val_loss: 0.5142 - val_acc: 0.78
83
Epoch 4/10
640000/640000 [==============================] - 119s 187us/step - loss: 0.6041 - acc: 0.6725 - val_loss: 0.5181 - val_acc: 0.78
00
Epoch 5/10
640000/640000 [==============================] - 119s 187us/step - loss: 0.5904 - acc: 0.6832 - val_loss: 0.5167 - val_acc: 0.77
32
Epoch 6/10
640000/640000 [==============================] - 119s 187us/step - loss: 0.5774 - acc: 0.6918 - val_loss: 0.5199 - val_acc: 0.76
89
Epoch 7/10
640000/640000 [==============================] - 119s 187us/step - loss: 0.5658 - acc: 0.6981 - val_loss: 0.5205 - val_acc: 0.76
29
Epoch 8/10
640000/640000 [==============================] - 119s 187us/step - loss: 0.5553 - acc: 0.7034 - val_loss: 0.5268 - val_acc: 0.76
09
Epoch 9/10
640000/640000 [==============================] - 119s 187us/step - loss: 0.5457 - acc: 0.7085 - val_loss: 0.5363 - val_acc: 0.74
94
Epoch 10/10
640000/640000 [==============================] - 119s 187us/step - loss: 0.5372 - acc: 0.7116 - val_loss: 0.5387 - val_acc: 0.75
09
80000/80000 [==============================] - 3s 35us/step
acc: 75.22%
```

sigmoid

In [17]:
```
# hidden_layer = sigmoid, l2norm = True, dropout = True
hidden_actfunc = "sigmoid"
l2normFlag = True
dropoutFlag = True
dropoutRate = 0.5
model = sequential_model (hidden_actfunc, l2normFlag, dropoutFlag, dropoutRate)
print("--------------------------------------------------------------------------------")
print("hidden_actfunc: "+hidden_actfunc, "  l2normFlag: " + str(l2normFlag),
      "  dropoutFlag: "+str(dropoutFlag), "  dropoutRate " + str(dropoutRate))
model.fit(train_data, train_target,batch_size=BATCH_SIZE,epochs=N_EPOCHS, validation_data =(val_data, val_target))
scores = model.evaluate(test_data, test_target)
print("%s: %.2f%%"%(model.metrics_names[1], scores[1]*100))
```

```
--------------------------------------------------------------------------------
hidden_actfunc: sigmoid   l2normFlag: True   dropoutFlag: True   dropoutRate 0.5
Train on 640000 samples, validate on 80000 samples
Epoch 1/10
640000/640000 [==============================] - 123s 192us/step - loss: 0.7359 - acc: 0.5033 - val_loss: 0.6999 - val_acc: 0.50
00
Epoch 2/10
640000/640000 [==============================] - 122s 190us/step - loss: 0.7000 - acc: 0.5011 - val_loss: 0.7003 - val_acc: 0.50
00
Epoch 3/10
640000/640000 [==============================] - 122s 190us/step - loss: 0.7003 - acc: 0.5030 - val_loss: 0.7016 - val_acc: 0.51
45
Epoch 4/10
640000/640000 [==============================] - 122s 190us/step - loss: 0.7029 - acc: 0.5089 - val_loss: 0.7070 - val_acc: 0.52
84
Epoch 5/10
640000/640000 [==============================] - 122s 190us/step - loss: 0.7200 - acc: 0.5501 - val_loss: 0.7070 - val_acc: 0.68
33
Epoch 6/10
640000/640000 [==============================] - 122s 190us/step - loss: 0.7268 - acc: 0.5788 - val_loss: 0.6925 - val_acc: 0.71
46
```

```
Epoch 7/10
640000/640000 [==============================] - 122s 190us/step - loss: 0.7260 - acc: 0.5922 - val_loss: 0.6787 - val_acc: 0.72
52
Epoch 8/10
640000/640000 [==============================] - 122s 190us/step - loss: 0.7245 - acc: 0.6002 - val_loss: 0.6744 - val_acc: 0.74
44
Epoch 9/10
640000/640000 [==============================] - 122s 190us/step - loss: 0.7225 - acc: 0.6058 - val_loss: 0.6714 - val_acc: 0.73
96
Epoch 10/10
640000/640000 [==============================] - 122s 190us/step - loss: 0.7208 - acc: 0.6105 - val_loss: 0.6606 - val_acc: 0.74
33
80000/80000 [==============================] - 3s 35us/step
acc: 74.14%
```

In [18]:
```
# hidden_layer = sigmoid, l2norm = True, dropout = False
hidden_actfunc = "sigmoid"
l2normFlag = True
dropoutFlag = False
dropoutRate = 0.5
model = sequential_model (hidden_actfunc, l2normFlag, dropoutFlag, dropoutRate)
print("--------------------------------------------------------------------------------")
print("hidden_actfunc: "+hidden_actfunc, "  l2normFlag: " + str(l2normFlag),
      "  dropoutFlag: "+str(dropoutFlag), "  dropoutRate " + str(dropoutRate))
model.fit(train_data, train_target,batch_size=BATCH_SIZE,epochs=N_EPOCHS, validation_data =(val_data, val_target))
scores = model.evaluate(test_data, test_target)
print("%s: %.2f%%"%(model.metrics_names[1], scores[1]*100))
```

```
--------------------------------------------------------------------------------
hidden_actfunc: sigmoid   l2normFlag: True   dropoutFlag: False   dropoutRate 0.5
Train on 640000 samples, validate on 80000 samples
Epoch 1/10
640000/640000 [==============================] - 121s 188us/step - loss: 0.7194 - acc: 0.5029 - val_loss: 0.6935 - val_acc: 0.50
00
Epoch 2/10
640000/640000 [==============================] - 120s 187us/step - loss: 0.6940 - acc: 0.5008 - val_loss: 0.6941 - val_acc: 0.50
00
Epoch 3/10
640000/640000 [==============================] - 120s 187us/step - loss: 0.6939 - acc: 0.4999 - val_loss: 0.6945 - val_acc: 0.50
00
Epoch 4/10
640000/640000 [==============================] - 120s 187us/step - loss: 0.6937 - acc: 0.5002 - val_loss: 0.6934 - val_acc: 0.50
00
Epoch 5/10
640000/640000 [==============================] - 120s 187us/step - loss: 0.6935 - acc: 0.5002 - val_loss: 0.6933 - val_acc: 0.50
00
Epoch 6/10
640000/640000 [==============================] - 120s 187us/step - loss: 0.6934 - acc: 0.4982 - val_loss: 0.6934 - val_acc: 0.50
00
Epoch 7/10
640000/640000 [==============================] - 120s 187us/step - loss: 0.6933 - acc: 0.4996 - val_loss: 0.6933 - val_acc: 0.50
00
Epoch 8/10
640000/640000 [==============================] - 120s 187us/step - loss: 0.6933 - acc: 0.5010 - val_loss: 0.6934 - val_acc: 0.50
00
Epoch 9/10
640000/640000 [==============================] - 120s 187us/step - loss: 0.6933 - acc: 0.5000 - val_loss: 0.6932 - val_acc: 0.50
00
Epoch 10/10
640000/640000 [==============================] - 120s 187us/step - loss: 0.6933 - acc: 0.4996 - val_loss: 0.6933 - val_acc: 0.50
00
80000/80000 [==============================] - 3s 35us/step
acc: 50.00%
```

In [19]:
```
# hidden_layer = sigmoid, l2norm = False, dropout = True
hidden_actfunc = "sigmoid"
l2normFlag = False
dropoutFlag = True
dropoutRate = 0.5
model = sequential_model (hidden_actfunc, l2normFlag, dropoutFlag, dropoutRate)
print("--------------------------------------------------------------------------------")
print("hidden_actfunc: "+hidden_actfunc, "  l2normFlag: " + str(l2normFlag),
      "  dropoutFlag: "+str(dropoutFlag), "  dropoutRate " + str(dropoutRate))
model.fit(train_data, train_target,batch_size=BATCH_SIZE,epochs=N_EPOCHS, validation_data =(val_data, val_target))
scores = model.evaluate(test_data, test_target)
print("%s: %.2f%%"%(model.metrics_names[1], scores[1]*100))
```

```
--------------------------------------------------------------------------------
hidden_actfunc: sigmoid   l2normFlag: False   dropoutFlag: True   dropoutRate 0.5
Train on 640000 samples, validate on 80000 samples
Epoch 1/10
640000/640000 [==============================] - 123s 192us/step - loss: 0.6655 - acc: 0.5938 - val_loss: 0.5595 - val_acc: 0.76
90
Epoch 2/10
640000/640000 [==============================] - 122s 190us/step - loss: 0.6477 - acc: 0.6240 - val_loss: 0.5368 - val_acc: 0.77
79
Epoch 3/10
640000/640000 [==============================] - 122s 190us/step - loss: 0.6418 - acc: 0.6330 - val_loss: 0.5381 - val_acc: 0.78
87
Epoch 4/10
640000/640000 [==============================] - 122s 190us/step - loss: 0.6371 - acc: 0.6405 - val_loss: 0.5258 - val_acc: 0.79
01
```

```
Epoch 5/10
640000/640000 [==============================] - 122s 190us/step - loss: 0.6336 - acc: 0.6443 - val_loss: 0.5293 - val_acc: 0.79
35
Epoch 6/10
640000/640000 [==============================] - 122s 190us/step - loss: 0.6311 - acc: 0.6473 - val_loss: 0.5252 - val_acc: 0.79
16
Epoch 7/10
640000/640000 [==============================] - 122s 190us/step - loss: 0.6284 - acc: 0.6504 - val_loss: 0.5248 - val_acc: 0.79
24
Epoch 8/10
640000/640000 [==============================] - 122s 190us/step - loss: 0.6262 - acc: 0.6533 - val_loss: 0.5080 - val_acc: 0.79
31
Epoch 9/10
640000/640000 [==============================] - 122s 190us/step - loss: 0.6239 - acc: 0.6558 - val_loss: 0.5055 - val_acc: 0.79
35
Epoch 10/10
640000/640000 [==============================] - 122s 190us/step - loss: 0.6220 - acc: 0.6580 - val_loss: 0.5058 - val_acc: 0.79
25
80000/80000 [==============================] - 3s 35us/step
acc: 79.44%
```

In [20]:
```
# hidden_layer = sigmoid, l2norm = False, dropout = False
hidden_actfunc = "sigmoid"
l2normFlag = False
dropoutFlag = False
dropoutRate = 0.5
model = sequential_model (hidden_actfunc, l2normFlag, dropoutFlag, dropoutRate)
print("-------------------------------------------------------------------------------")
print("hidden_actfunc: "+hidden_actfunc, "  l2normFlag: " + str(l2normFlag),
      "  dropoutFlag: "+str(dropoutFlag), "  dropoutRate " + str(dropoutRate))
model.fit(train_data, train_target,batch_size=BATCH_SIZE,epochs=N_EPOCHS, validation_data =(val_data, val_target))
scores = model.evaluate(test_data, test_target)
print("%s: %.2f%%"%(model.metrics_names[1], scores[1]*100))
```

```
-------------------------------------------------------------------------------
hidden_actfunc: sigmoid   l2normFlag: False   dropoutFlag: False   dropoutRate 0.5
Train on 640000 samples, validate on 80000 samples
Epoch 1/10
640000/640000 [==============================] - 120s 188us/step - loss: 0.6509 - acc: 0.6166 - val_loss: 0.5399 - val_acc: 0.77
65
Epoch 2/10
640000/640000 [==============================] - 119s 186us/step - loss: 0.6301 - acc: 0.6468 - val_loss: 0.5184 - val_acc: 0.78
41
Epoch 3/10
640000/640000 [==============================] - 119s 186us/step - loss: 0.6192 - acc: 0.6591 - val_loss: 0.5174 - val_acc: 0.78
87
Epoch 4/10
640000/640000 [==============================] - 119s 186us/step - loss: 0.6091 - acc: 0.6690 - val_loss: 0.5077 - val_acc: 0.78
55
Epoch 5/10
640000/640000 [==============================] - 119s 186us/step - loss: 0.5989 - acc: 0.6780 - val_loss: 0.5158 - val_acc: 0.77
92
Epoch 6/10
640000/640000 [==============================] - 119s 186us/step - loss: 0.5887 - acc: 0.6863 - val_loss: 0.5116 - val_acc: 0.77
51
Epoch 7/10
640000/640000 [==============================] - 119s 186us/step - loss: 0.5786 - acc: 0.6934 - val_loss: 0.5154 - val_acc: 0.76
92
Epoch 8/10
640000/640000 [==============================] - 119s 186us/step - loss: 0.5688 - acc: 0.6997 - val_loss: 0.5226 - val_acc: 0.76
52
Epoch 9/10
640000/640000 [==============================] - 119s 186us/step - loss: 0.5599 - acc: 0.7046 - val_loss: 0.5223 - val_acc: 0.75
95
Epoch 10/10
640000/640000 [==============================] - 119s 186us/step - loss: 0.5516 - acc: 0.7089 - val_loss: 0.5329 - val_acc: 0.75
32
80000/80000 [==============================] - 3s 35us/step
acc: 75.52%
```

tanh

In [21]:
```
# hidden_layer = tanh, l2norm = True, dropout = True
hidden_actfunc = "sigmoid"
l2normFlag = True
dropoutFlag = True
dropoutRate = 0.5
model = sequential_model (hidden_actfunc, l2normFlag, dropoutFlag, dropoutRate)
print("-------------------------------------------------------------------------------")
print("hidden_actfunc: "+hidden_actfunc, "  l2normFlag: " + str(l2normFlag),
      "  dropoutFlag: "+str(dropoutFlag), "  dropoutRate " + str(dropoutRate))
model.fit(train_data, train_target,batch_size=BATCH_SIZE,epochs=N_EPOCHS, validation_data =(val_data, val_target))
scores = model.evaluate(test_data, test_target)
print("%s: %.2f%%"%(model.metrics_names[1], scores[1]*100))
```

```
-------------------------------------------------------------------------------
hidden_actfunc: sigmoid   l2normFlag: True   dropoutFlag: True   dropoutRate 0.5
Train on 640000 samples, validate on 80000 samples
Epoch 1/10
640000/640000 [==============================] - 122s 191us/step - loss: 0.7307 - acc: 0.5031 - val_loss: 0.6988 - val_acc: 0.50
```

```
00
Epoch 2/10
640000/640000 [==============================] - 121s 189us/step - loss: 0.6999 - acc: 0.5010 - val_loss: 0.7013 - val_acc: 0.50
00
Epoch 3/10
640000/640000 [==============================] - 123s 192us/step - loss: 0.7000 - acc: 0.5019 - val_loss: 0.7002 - val_acc: 0.51
17
Epoch 4/10
640000/640000 [==============================] - 125s 195us/step - loss: 0.7027 - acc: 0.5085 - val_loss: 0.6984 - val_acc: 0.62
01
Epoch 5/10
640000/640000 [==============================] - 124s 195us/step - loss: 0.7191 - acc: 0.5481 - val_loss: 0.7060 - val_acc: 0.66
66
Epoch 6/10
640000/640000 [==============================] - 128s 200us/step - loss: 0.7268 - acc: 0.5780 - val_loss: 0.6975 - val_acc: 0.71
42
Epoch 7/10
640000/640000 [==============================] - 131s 204us/step - loss: 0.7261 - acc: 0.5914 - val_loss: 0.6838 - val_acc: 0.72
85
Epoch 8/10
640000/640000 [==============================] - 129s 202us/step - loss: 0.7243 - acc: 0.5994 - val_loss: 0.6692 - val_acc: 0.74
10
Epoch 9/10
640000/640000 [==============================] - 129s 201us/step - loss: 0.7228 - acc: 0.6055 - val_loss: 0.6675 - val_acc: 0.74
57
Epoch 10/10
640000/640000 [==============================] - 130s 204us/step - loss: 0.7213 - acc: 0.6102 - val_loss: 0.6573 - val_acc: 0.74
83
80000/80000 [==============================] - 3s 35us/step
acc: 74.62%
```

In [22]:

```
# hidden_layer = tanh, l2norm = True, dropout = False
hidden_actfunc = "tanh"
l2normFlag = True
dropoutFlag = False
dropoutRate = 0.5
model = sequential_model (hidden_actfunc, l2normFlag, dropoutFlag, dropoutRate)
print("--------------------------------------------------------------------------------")
print("hidden_actfunc: "+hidden_actfunc, "  l2normFlag: " + str(l2normFlag),
      "  dropoutFlag: "+str(dropoutFlag), "  dropoutRate " + str(dropoutRate))
model.fit(train_data, train_target,batch_size=BATCH_SIZE,epochs=N_EPOCHS, validation_data =(val_data, val_target))
scores = model.evaluate(test_data, test_target)
print("%s: %.2f%%"%(model.metrics_names[1], scores[1]*100))
```

```
--------------------------------------------------------------------------------
hidden_actfunc: tanh   l2normFlag: True   dropoutFlag: False   dropoutRate 0.5
Train on 640000 samples, validate on 80000 samples
Epoch 1/10
640000/640000 [==============================] - 128s 200us/step - loss: 0.7583 - acc: 0.5656 - val_loss: 0.6652 - val_acc: 0.70
44
Epoch 2/10
640000/640000 [==============================] - 128s 199us/step - loss: 0.7078 - acc: 0.6079 - val_loss: 0.6407 - val_acc: 0.73
16
Epoch 3/10
640000/640000 [==============================] - 131s 205us/step - loss: 0.7031 - acc: 0.6206 - val_loss: 0.6365 - val_acc: 0.74
47
Epoch 4/10
640000/640000 [==============================] - 129s 202us/step - loss: 0.6979 - acc: 0.6281 - val_loss: 0.6186 - val_acc: 0.76
37
Epoch 5/10
640000/640000 [==============================] - 129s 202us/step - loss: 0.6940 - acc: 0.6330 - val_loss: 0.5977 - val_acc: 0.76
97
Epoch 6/10
640000/640000 [==============================] - 130s 202us/step - loss: 0.6910 - acc: 0.6370 - val_loss: 0.6087 - val_acc: 0.77
99
Epoch 7/10
640000/640000 [==============================] - 129s 202us/step - loss: 0.6883 - acc: 0.6390 - val_loss: 0.5953 - val_acc: 0.77
64
Epoch 8/10
640000/640000 [==============================] - 129s 202us/step - loss: 0.6866 - acc: 0.6411 - val_loss: 0.5900 - val_acc: 0.77
94
Epoch 9/10
640000/640000 [==============================] - 129s 202us/step - loss: 0.6843 - acc: 0.6431 - val_loss: 0.6010 - val_acc: 0.78
30
Epoch 10/10
640000/640000 [==============================] - 129s 202us/step - loss: 0.6828 - acc: 0.6450 - val_loss: 0.5938 - val_acc: 0.77
61
80000/80000 [==============================] - 3s 35us/step
acc: 77.58%
```

In [27]:

```
# hidden_layer = tanh, l2norm = False, dropout = True
hidden_actfunc = "tanh"
l2normFlag = False
dropoutFlag = True
dropoutRate = 0.5
model = sequential_model (hidden_actfunc, l2normFlag, dropoutFlag, dropoutRate)
print("--------------------------------------------------------------------------------")
print("hidden_actfunc: "+hidden_actfunc, "  l2normFlag: " + str(l2normFlag),
      "  dropoutFlag: "+str(dropoutFlag), "  dropoutRate " + str(dropoutRate))
model.fit(train_data, train_target,batch_size=BATCH_SIZE,epochs=N_EPOCHS, validation_data =(val_data, val_target))
scores = model.evaluate(test_data, test_target)
print("%s: %.2f%%"%(model.metrics_names[1], scores[1]*100))
```

```
--------------------------------------------------------------------------------
hidden_actfunc: tanh   l2normFlag: False   dropoutFlag: True   dropoutRate 0.5
Train on 640000 samples, validate on 80000 samples
Epoch 1/10
640000/640000 [==============================] - 130s 203us/step - loss: 0.6764 - acc: 0.5847 - val_loss: 0.5582 - val_acc: 0.76
03
Epoch 2/10
640000/640000 [==============================] - 131s 205us/step - loss: 0.6514 - acc: 0.6180 - val_loss: 0.5410 - val_acc: 0.77
80
Epoch 3/10
640000/640000 [==============================] - 131s 205us/step - loss: 0.6444 - acc: 0.6293 - val_loss: 0.5231 - val_acc: 0.78
45
Epoch 4/10
640000/640000 [==============================] - 131s 205us/step - loss: 0.6406 - acc: 0.6353 - val_loss: 0.5256 - val_acc: 0.79
04
Epoch 5/10
640000/640000 [==============================] - 131s 205us/step - loss: 0.6376 - acc: 0.6396 - val_loss: 0.5248 - val_acc: 0.79
15
Epoch 6/10
640000/640000 [==============================] - 131s 205us/step - loss: 0.6353 - acc: 0.6429 - val_loss: 0.5182 - val_acc: 0.79
29
Epoch 7/10
640000/640000 [==============================] - 131s 205us/step - loss: 0.6326 - acc: 0.6467 - val_loss: 0.5159 - val_acc: 0.79
43
Epoch 8/10
640000/640000 [==============================] - 131s 205us/step - loss: 0.6312 - acc: 0.6489 - val_loss: 0.5213 - val_acc: 0.79
46
Epoch 9/10
640000/640000 [==============================] - 131s 205us/step - loss: 0.6294 - acc: 0.6509 - val_loss: 0.5217 - val_acc: 0.79
65
Epoch 10/10
640000/640000 [==============================] - 123s 192us/step - loss: 0.6279 - acc: 0.6533 - val_loss: 0.5091 - val_acc: 0.79
64
80000/80000 [==============================] - 3s 35us/step
acc: 79.75%
```

In [28]:
```
# hidden_layer = tanh, l2norm = False, dropout = False
hidden_actfunc = "tanh"
l2normFlag = False
dropoutFlag = False
dropoutRate = 0.5
model = sequential_model (hidden_actfunc, l2normFlag, dropoutFlag, dropoutRate)
print("--------------------------------------------------------------------------------")
print("hidden_actfunc: "+hidden_actfunc, "  l2normFlag: " + str(l2normFlag),
      "  dropoutFlag: "+str(dropoutFlag), "  dropoutRate " + str(dropoutRate))
model.fit(train_data, train_target,batch_size=BATCH_SIZE,epochs=N_EPOCHS, validation_data =(val_data, val_target))
scores = model.evaluate(test_data, test_target)
print("%s: %.2f%%"%(model.metrics_names[1], scores[1]*100))
```

```
--------------------------------------------------------------------------------
hidden_actfunc: tanh   l2normFlag: False   dropoutFlag: False   dropoutRate 0.5
Train on 640000 samples, validate on 80000 samples
Epoch 1/10
640000/640000 [==============================] - 124s 194us/step - loss: 0.6568 - acc: 0.6089 - val_loss: 0.5386 - val_acc: 0.76
88
Epoch 2/10
640000/640000 [==============================] - 123s 192us/step - loss: 0.6357 - acc: 0.6392 - val_loss: 0.5306 - val_acc: 0.77
86
Epoch 3/10
640000/640000 [==============================] - 122s 191us/step - loss: 0.6270 - acc: 0.6500 - val_loss: 0.5222 - val_acc: 0.78
29
Epoch 4/10
640000/640000 [==============================] - 123s 193us/step - loss: 0.6206 - acc: 0.6579 - val_loss: 0.5210 - val_acc: 0.78
44
Epoch 5/10
640000/640000 [==============================] - 134s 210us/step - loss: 0.6146 - acc: 0.6636 - val_loss: 0.5155 - val_acc: 0.78
08
Epoch 6/10
640000/640000 [==============================] - 135s 211us/step - loss: 0.6084 - acc: 0.6699 - val_loss: 0.5179 - val_acc: 0.77
86
Epoch 7/10
640000/640000 [==============================] - 129s 202us/step - loss: 0.6025 - acc: 0.6760 - val_loss: 0.5183 - val_acc: 0.77
78
Epoch 8/10
640000/640000 [==============================] - 129s 201us/step - loss: 0.5963 - acc: 0.6811 - val_loss: 0.5136 - val_acc: 0.77
29
Epoch 9/10
640000/640000 [==============================] - 129s 202us/step - loss: 0.5905 - acc: 0.6854 - val_loss: 0.5143 - val_acc: 0.77
24
Epoch 10/10
640000/640000 [==============================] - 129s 202us/step - loss: 0.5842 - acc: 0.6903 - val_loss: 0.5130 - val_acc: 0.76
79
80000/80000 [==============================] - 3s 35us/step
acc: 76.59%
```