0.4 Convolutional Neural Network (CNN) model

```
In [0]: # define new callback for ROC AUC score for CNN
        class roc_callback(Callback):
            def __init__(self,training_data,validation_data):
                self.x = training_data[0]
                self.y = training_data[1]
                self.x_val = validation_data[0]
                self.y_val = validation_data[1]

            def on_epoch_end(self, epoch, logs={}):
                y_pred = self.model.predict(self.x)
                roc = roc_auc_score(self.y, y_pred)

                y_pred_val = self.model.predict(self.x_val)
                roc_val = roc_auc_score(self.y_val, y_pred_val)

                print('\rroc-auc: %s - roc-auc_val: %s' % (str(round(roc,4)),str(round(roc_val
                return
```

```
In [0]: # function that compile, train and evaluate model
        def train_model(model,X_train,y_train,X_val,y_val,batch_size,epochs,filepath):
            # compile the model
            model.compile(optimizer='adam',
                          loss='binary_crossentropy',
                          metrics=['accuracy'])
            # parameters
            batch_size = batch_size
            epochs = epochs
            # callbacks
            checkpointer =  ModelCheckpoint(filepath=filepath,
                                            verbose=1,
                                            save_best_only=True)
            roc = roc_callback(training_data=(X_train,y_train),validation_data=(X_val,y_val))
            # fit the model
            history = model.fit(X_train,y_train,
                                validation_data=(X_val,y_val),
                                batch_size=batch_size,
                                epochs=epochs,
                                callbacks=[roc,checkpointer])
            # load the model with the best validation loss
            model.load_weights(filepath)
            return model, history
```

```
In [0]: def history_plot(history,filename):
            fig = plt.figure(figsize=(15, 5))
            # summarize history for accuracy
            plt.subplot(1, 2, 1)
            plt.plot(history.history['acc']); plt.plot(history.history['val_acc']);
```

```
plt.title('model accuracy'); plt.ylabel('accuracy');
plt.xlabel('epoch'); plt.legend(['train', 'valid'], loc='upper left');

# summarize history for loss
plt.subplot(1, 2, 2)
plt.plot(history.history['loss']); plt.plot(history.history['val_loss']);
plt.title('model loss'); plt.ylabel('loss');
plt.xlabel('epoch'); plt.legend(['train', 'valid'], loc='upper left');
plt.show()

fig.savefig(filename,bbox_inches = 'tight')
```

1 Model Architecture CNN

```
In [0]: max_features = 20000
        max_len=250
        # build CNN model
        model_cnn = Sequential()
        model_cnn.add(Embedding(max_features, 100, input_length=max_len))
        model_cnn.add(Conv1D(filters=16, kernel_size=3, activation='relu'))
        model_cnn.add(MaxPooling1D(pool_size=2))
        model_cnn.add(Dropout(0.5))
        model_cnn.add(Conv1D(filters=32, kernel_size=3, activation='relu'))
        model_cnn.add(MaxPooling1D(pool_size=2))
        model_cnn.add(Dropout(0.5))
        model_cnn.add(Flatten())
        model_cnn.add(Dense(600, activation="relu"))
        model_cnn.add(Dropout(0.5))
        model_cnn.add(Dense(6, activation="sigmoid"))

        #summary
        model_cnn.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_4 (Embedding)      (None, 250, 100)          2000000
_________________________________________________________________
conv1d_7 (Conv1D)            (None, 248, 16)           4816
_________________________________________________________________
max_pooling1d_7 (MaxPooling1 (None, 124, 16)           0
_________________________________________________________________
dropout_10 (Dropout)         (None, 124, 16)           0
_________________________________________________________________
conv1d_8 (Conv1D)            (None, 122, 32)           1568
_________________________________________________________________
max_pooling1d_8 (MaxPooling1 (None, 61, 32)            0
```

```
_________________________________________________________________
dropout_11 (Dropout)         (None, 61, 32)            0
_________________________________________________________________
flatten_4 (Flatten)          (None, 1952)              0
_________________________________________________________________
dense_7 (Dense)              (None, 600)               1171800
_________________________________________________________________
dropout_12 (Dropout)         (None, 600)               0
_________________________________________________________________
dense_8 (Dense)              (None, 6)                 3606
=================================================================
Total params: 3,181,790
Trainable params: 3,181,790
Non-trainable params: 0
_________________________________________________________________
```

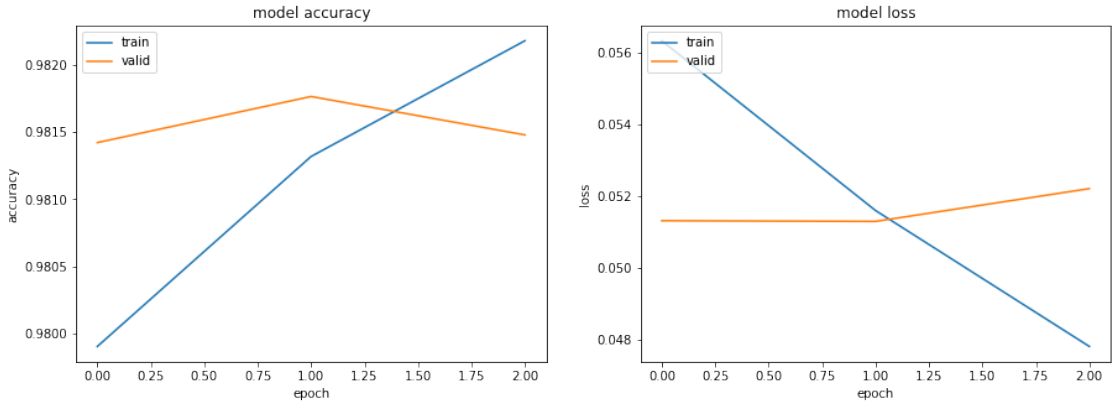
1.0.1 Train Model

```
In [0]: X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, train_size=0.9, ra

        model_cnn_trained, history_cnn = train_model(model_cnn,
                                                     X_train,y_train,
                                                     X_val,y_val,
                                                     batch_size=512,
                                                     epochs=3,
                                                     filepath='models/weights.best.from_scatch
```

```
Train on 104716 samples, validate on 11636 samples
Epoch 1/3
104716/104716 [==============================] - 61s 580us/step - loss: 0.0563 - acc: 0.9799 -
roc-auc: 0.9804 - roc-auc_val: 0.9766

Epoch 00001: val_loss improved from inf to 0.05130, saving model to models/weights.best.from_s
Epoch 2/3
104716/104716 [==============================] - 60s 571us/step - loss: 0.0516 - acc: 0.9813 -
roc-auc: 0.9829 - roc-auc_val: 0.9756

Epoch 00002: val_loss improved from 0.05130 to 0.05129, saving model to models/weights.best.fr
Epoch 3/3
104716/104716 [==============================] - 60s 569us/step - loss: 0.0478 - acc: 0.9822 -
roc-auc: 0.9846 - roc-auc_val: 0.9746

Epoch 00003: val_loss did not improve from 0.05129
```

1.0.2 Evaluation Plot

```
In [0]: history_plot(history_cnn,'history_cnn.eps')
```




```
In [0]: plot_model(model_cnn, to_file='figure/model_cnn.eps', show_shapes=True, show_layer_name
```

2 Transfer Learning (GloVe) model

```
In [0]: # load the whole embedding into memory
        def create_embeddings_index(filename):
            embeddings_index = dict()
            f = open(filename, encoding = 'utf-8')
            for line in f:
                values = line.split()
                word = values[0]
                coefs = np.asarray(values[1:], dtype='float32')
                embeddings_index[word] = coefs
            f.close()
            print('Loaded %s word vectors.' % len(embeddings_index.keys()))
            return embeddings_index
```

```
In [0]: # create a weight matrix for words in training docs (look up word vectors in embedding
        def create_embeddings_matrix(tokenizer, embeddings_index):
            vocab_size = len(tokenizer.word_index) + 1
            embeddings_matrix = np.zeros((vocab_size, 100))
            for word, i in tokenizer.word_index.items():
                embeddings_vector = embeddings_index.get(word)
                if embeddings_vector is not None:
                    embeddings_matrix[i] = embeddings_vector
            return embeddings_matrix
```

2.1 Create Embedding Matrix

```
In [0]: # GloVe model
        # load the whole embedding into memory
        embeddings_index_glove = create_embeddings_index('glove.6B.100d.txt')
```

```
        # create a weight matrix for the embedding layer from a loaded embedding
        embeddings_matrix_glove = create_embeddings_matrix(tokenizer, embeddings_index_glove)
```

```
Loaded 400000 word vectors.
```

2.2 Model Architecture : GloVe + RNN

```
In [0]: # define the model
        vocab_size = len(tokenizer.word_index) + 1

        model_glove = Sequential()
        model_glove.add(Embedding(vocab_size, 100, weights = [embeddings_matrix_glove], input_l
        model_glove.add(Bidirectional(LSTM(128, return_sequences=True)))
        model_glove.add(GlobalMaxPool1D())
        model_glove.add(Dropout(0.25))
        model_glove.add(Dense(60, activation="relu"))
        model_glove.add(Dropout(0.25))
        model_glove.add(Dense(6, activation="sigmoid"))

        # summary
        model_glove.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_5 (Embedding)      (None, 250, 100)          19090800
_________________________________________________________________
bidirectional_1 (Bidirection (None, 250, 256)          234496
_________________________________________________________________
global_max_pooling1d_1 (Glob (None, 256)               0
_________________________________________________________________
dropout_13 (Dropout)         (None, 256)               0
_________________________________________________________________
dense_9 (Dense)              (None, 60)                15420
_________________________________________________________________
dropout_14 (Dropout)         (None, 60)                0
_________________________________________________________________
dense_10 (Dense)             (None, 6)                 366
=================================================================
Total params: 19,341,082
Trainable params: 250,282
Non-trainable params: 19,090,800
_________________________________________________________________
```

2.3 Train Model

```
In [0]: model_glove_trained, history_glove = train_model(model_glove,
                                                 X_train,y_train,
                                                 X_val,y_val,
                                                 batch_size=512,
                                                 epochs=5,
                                                 filepath='models/weights.best.glove.hdf5')
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow/python/ops/math_grad
Instructions for updating:
Deprecated in favor of operator or tf.math.divide.
Train on 104716 samples, validate on 11636 samples
Epoch 1/5
104716/104716 [==============================] - 810s 8ms/step - loss: 0.1138 - acc: 0.9665 - v
roc-auc: 0.9598 - roc-auc_val: 0.9624

Epoch 00001: val_loss improved from inf to 0.06193, saving model to models/weights.best.glove.h
Epoch 2/5
104716/104716 [==============================] - 812s 8ms/step - loss: 0.0637 - acc: 0.9786 - v
roc-auc: 0.9678 - roc-auc_val: 0.9687

Epoch 00002: val_loss improved from 0.06193 to 0.05762, saving model to models/weights.best.glo
Epoch 3/5
104716/104716 [==============================] - 812s 8ms/step - loss: 0.0588 - acc: 0.9799 - v
roc-auc: 0.9725 - roc-auc_val: 0.971

Epoch 00003: val_loss improved from 0.05762 to 0.05640, saving model to models/weights.best.glo
Epoch 4/5
104716/104716 [==============================] - 819s 8ms/step - loss: 0.0559 - acc: 0.9803 - v
roc-auc: 0.976 - roc-auc_val: 0.9732

Epoch 00004: val_loss improved from 0.05640 to 0.05540, saving model to models/weights.best.glo
Epoch 5/5
104716/104716 [==============================] - 812s 8ms/step - loss: 0.0533 - acc: 0.9809 - v
roc-auc: 0.9793 - roc-auc_val: 0.9757

Epoch 00005: val_loss improved from 0.05540 to 0.05262, saving model to models/weights.best.glo
```

```
In [0]: # more training based on the previous model training
        model_glove_trained_2, history_glove_2 = train_model(model_glove_trained,
                                                             X_train,y_train,
                                                             X_val,y_val,
                                                             batch_size=512,
                                                             epochs=5,
                                                             filepath='models/weights.best.glov
```

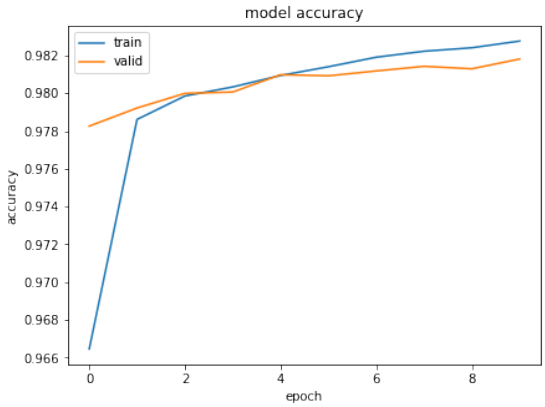
```
Train on 104716 samples, validate on 11636 samples
Epoch 1/5
```

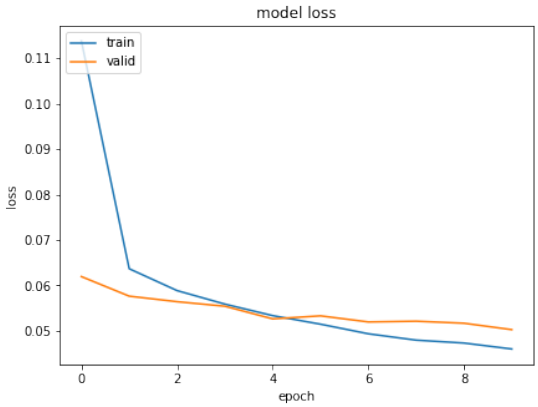
```
104716/104716 [==============================] - 813s 8ms/step - loss: 0.0514 - acc: 0.9814 - v
roc-auc: 0.9816 - roc-auc_val: 0.9748

Epoch 00001: val_loss improved from inf to 0.05328, saving model to models/weights.best.glove_2
Epoch 2/5
104716/104716 [==============================] - 813s 8ms/step - loss: 0.0493 - acc: 0.9819 - v
roc-auc: 0.9835 - roc-auc_val: 0.9766

Epoch 00002: val_loss improved from 0.05328 to 0.05194, saving model to models/weights.best.glo
Epoch 3/5
104716/104716 [==============================] - 808s 8ms/step - loss: 0.0479 - acc: 0.9822 - v
roc-auc: 0.9848 - roc-auc_val: 0.9776

Epoch 00003: val_loss did not improve from 0.05194
Epoch 4/5
104716/104716 [==============================] - 810s 8ms/step - loss: 0.0473 - acc: 0.9824 - v
roc-auc: 0.9858 - roc-auc_val: 0.978

Epoch 00004: val_loss improved from 0.05194 to 0.05166, saving model to models/weights.best.glo
Epoch 5/5
104716/104716 [==============================] - 812s 8ms/step - loss: 0.0460 - acc: 0.9828 - v
roc-auc: 0.9873 - roc-auc_val: 0.9794

Epoch 00005: val_loss improved from 0.05166 to 0.05025, saving model to models/weights.best.glo
```

2.4 Evaluation Plot

```
In [0]: plot_model(model_glove, to_file='figure/model_glove.eps', show_shapes=True, show_layer_
```

```
In [0]: fig = plt.figure(figsize=(15, 5))
        # summarize history for accuracy
        plt.subplot(1, 2, 1)
        plt.plot(history_glove.history['acc']+history_glove_2.history['acc']);
        plt.plot(history_glove.history['val_acc']+history_glove_2.history['val_acc']);
        plt.title('model accuracy'); plt.ylabel('accuracy');
        plt.xlabel('epoch'); plt.legend(['train', 'valid'], loc='upper left');

        # summarize history for loss
        plt.subplot(1, 2, 2)
        plt.plot(history_glove.history['loss']+history_glove_2.history['loss']);
        plt.plot(history_glove.history['val_loss']+history_glove_2.history['val_loss']);
        plt.title('model loss'); plt.ylabel('loss');
        plt.xlabel('epoch'); plt.legend(['train', 'valid'], loc='upper left');
        plt.show()

        fig.savefig('figure/history_glove.eps',bbox_inches = 'tight')
```


model accuracy
train
valid
accuracy
epoch
0.982
0.980
0.978
0.976
0.974
0.972
0.970
0.968
0.966
0
2
4
6
8


model loss
train
valid
loss
epoch
0.11
0.10
0.09
0.08
0.07
0.06
0.05
0
2
4
6
8