Abstract:

How does the brain solve visual object recognition? This study attempts to address this question and modeling the visual object recognition of human beings. For do this, we propose an effective representations from fMRI data by deep learning based classifier.

1. Description of Applied Problem:

Humans show excellent performance in interpreting visual scenes and detect and classify objects from among tens of thousands of possibilities which is still unreachable by machines. Effective representations from fMRI data is important in Brain Computer Interfaces (BCI) and transferring human visual capability to computer vision methods.

For modeling the visual object recognition of human beings, we propose fMRI data extract by visual object stimuli combined with Recurrent Neural Networks (RNN) to learn a discriminative brain activity of visual categories.

2. Description of Available Data:

BOLD 5000 is dramatically increased the stimulus set size in an fMRI study of visual scene processing. In this data set, 4 participants is scanned in a slow-evented related design with 4,916 unique scenes. Data was collected over 16 sessions, 15 of which were task-related sessions, plus an additional session for acquiring high resolution anatomical scans.

BOLD 5000 incorporates the two core values of neuroscience and computer vision into one dataset. From neuroscience, a careful setup of slow event-related is taken to ensure a stimulus to response mapping. From computer vision, some of the most standard benchmark images is taken as indoor/outdoor scenes and a lot of naturalistic images. The stimuli were drawn from three datasets: 1) Scene Images 2) COCO dataset 3) ImageNet dataset.

Images were presented for 1 second, with 9 seconds of fixation between trials. Participants were asked to judge whether they liked, disliked, or were neutral about the image.

The images in ImageNet dataset are labeled with ImageNet super categories. These super categories were created by using the WordNet hierarchy. From 61 final WordNet categories, images are labeled as *"**Living Animate**", "**Living Inanimate**", "**Objects**", "**Food**"*or *"**Geography**"*. An example of label mapping is "Dog" to "Living Animate" and "Vehicle" to "Objects".

3. Analysis Techniques.

Convolutional and recurrent neural networks have achieved great success in many extracting representations from fMRI data, such as sleep stage classification, motor imagery and Alzheimer's disease recognition . Thus, the contribution of our work is :

Preprocessing: In Bold 5000 data set, 10 regions of interest (ROIs) are defined from the localizer and then the data of the ROIs were extracted across the timecourse of the trial presentation (TR1 = 0-2s, TR2 = 2-4s, TR3 = 4-6s, TR4 = 6-8s, TR5 = 8-10s). We analyzed the data for each timepoint by using PCA method to dimensional reduction and improve the performance of classification. In this work, for each time step 30 first PCAs are extracted. In the other words experiments are represented by using a feature vector of size 150.

Analysis : We propose a deep learning based approach to classify image category fMRI Dataset by outperforming state-of-the-art methods in time-series classification accuracy.

Classic methods of Machine learning ignore the temporal dynamics of fMRI which accounts as the most important feature of fMRI data. Using deep learning and recurrent neural network would be a solution for such situations. For this purpose, we applied LSTM recurrent neural networks, which is more commonly used than simple recurrent neural network due to their capability of tracking long-term dynamics of fMRI time series data. Stacked LSTM networks has been implemented using keras library (TensorFlow backend). The last layer of network is a Dense layer using softmax activation function.

Evaluation and Visualization:

To explore and make sense of results we used confusion matrix and and ROC curve.

```
X_train (1842, 5, 30), Y_train = (1842, 1), X_test (205, 5, 30)

Class labels: #1 food(20), #2 inanimate(24), #3 animated(780),
 #4 objects(1113), #5 geo(20)
```

Parameters:

```
model.add(LSTM(500, input_shape=(5, 30), return_sequences=True,
activation='relu', dropout=0.7))

model.add(LSTM(400,activation='tanh',dropout=0.5,return_sequences=True))

model.add(LSTM(300,activation='tanh',dropout=0.6))

model.add(Dense(5, activation="softmax")) # number of classes 5,

Adam =optimizers.adam(lr=0.001,beta_1=0.9, beta_2=0.999)
model.compile(loss='categorical_crossentropy', optimizer=Adam,
metrics=['accuracy'])

batch_size = 50
epochs = 50
```

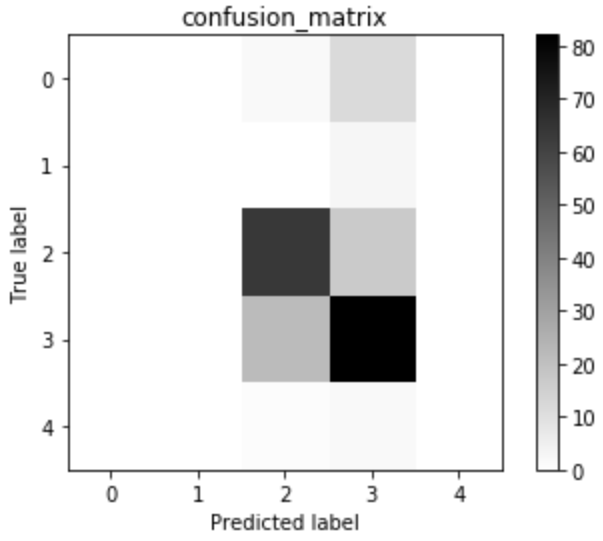
Results:

```
Epoch 00035: val_acc did not improve from 0.71351

Epoch 00036: val_acc improved from 0.71351 to 0.71351, saving model to
weights-improvement-36-0.71.hdf5

Epoch 00037: val_acc did not improve from 0.71351
.
.
.
Epoch 00049: val_acc did not improve from 0.71351

Epoch 00050: val_acc did not improve from 0.71351 -----> validation_split=0.1
                                          acc: 71.22% ----> Test set
```

	precision	recall	f1-score	support
0.0	0.00	0.00	0.00	14
1.0	0.00	0.00	0.00	3
2.0	0.72	0.79	0.75	81
3.0	0.71	0.79	0.75	104
4.0	0.00	0.00	0.00	3
micro avg	0.71	0.71	0.71	205
macro avg	0.29	0.32	0.30	205
weighted avg	0.64	0.71	0.68	205

confusion_matrix

```
[[ 0  0  2 12  0]
 [ 0  0  0  3  0]
 [ 0  0 64 17  0]
 [ 0  0 22 82  0]
 [ 0  0  1  2  0]]
```



Support Vector Machine:

1) Linear SVM
2) kernel=poly with degree=8
3) kernel=RBF (Gaussian)
4) kernel=sigmoid

Accuracy score:

0.7170731707317073

0.5902439024390244

0.5902439024390244

0.5902439024390244

Results of Linear SVM

	precision	recall	f1-score	support
1	0.00	0.00	0.00	13
2	0.00	0.00	0.00	5
3	0.65	0.69	0.67	64
4	0.75	0.85	0.80	121
5	0.00	0.00	0.00	2
micro avg	0.72	0.72	0.72	205
macro avg	0.28	0.31	0.29	205
weighted avg	0.65	0.72	0.68	205

Confusion matrix of linear SVM

```
[[  0   0   4   9   0]
 [  0   0   1   4   0]
 [  0   0  44  20   0]
 [  0   0  18 103   0]
 [  0   0   1   1   0]]
```

Two other experiments on LSTM:

Parameters...

```
model.add(LSTM(500, input_shape=(5, 30), return_sequences=True,  activation='relu',    dropout=0.7))
 # returns a sequence of vectors of dimension 500
 model.add(LSTM(300,activation='tanh',dropout=0.5,return_sequences=True))
 # returns a sequence of vectors of dimension 400
model.add(LSTM(100,activation='tanh',dropout=0.3))
 # return a single vector of dimension 200
model.add(Dense(5, activation="softmax")) # number of classes:  5,

Adam =optimizers.adam(lr=0.001,beta_1=0.9, beta_2=0.999)
model.compile(loss='categorical_crossentropy', optimizer=Adam, metrics=['accuracy'])

batch_size = 50.
epochs = 50
```

Results….

Epoch 00024: val_acc improved from 0.69730 to 0.71892, saving model to weights-improvement-24-0.72.hdf5

Epoch 00025: val_acc improved from 0.71892 to 0.72973, saving model to weights-improvement-25-0.73.hdf5

Epoch 00026: val_acc improved from 0.72973 to 0.75676, saving model to weights-improvement-26-0.76.hdf5

Epoch 00027: val_acc did not improve from 0.75676

Epoch 00028: val_acc did not improve from 0.75676

Epoch 00029: val_acc did not improve from 0.75676

.

.

.

Epoch 00049: val_acc did not improve from 0.75676

Epoch 00050: val_acc did not improve from 0.75676 -----> validation_split=0.1

acc: 72.20% **-----> Test set**

Parameters:

```
model.add(LSTM(500, input_shape=(5, 30), return_sequences=True,  activation='relu', dropout=0.7))
  # returns a sequence of vectors of dimension 500
  model.add(LSTM(400,activation='tanh',dropout=0.6,return_sequences=True))
  # returns a sequence of vectors of dimension 400
  model.add(LSTM(200,activation='tanh',dropout=0.4))
  # return a single vector of dimension 200
  model.add(Dense(5, activation="softmax")) # number of classes 5,.

  Adam =optimizers.adam(lr=0.001,beta_1=0.9, beta_2=0.999)
  model.compile(loss='categorical_crossentropy', optimizer=Adam, metrics=['accuracy'])
  return model

batch_size = 50
epochs = 50
```

Results:

Epoch 00040: val_acc improved from 0.74054 to 0.75676, saving model to weights-improvement-40-0.76.hdf5

Epoch 00041: val_acc did not improve from 0.75676

Epoch 00042: val_acc improved from 0.75676 to 0.76757, saving model to weights-improvement-42-0.77.hdf5

Epoch 00043: val_acc did not improve from 0.76757

.

.

.

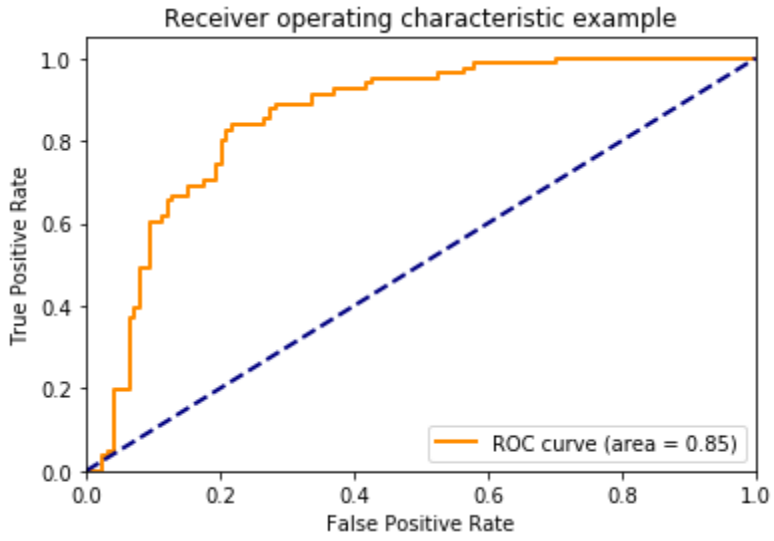
Epoch 00048: val_acc did not improve from 0.76757

Epoch 00049: val_acc improved from 0.76757 to 0.76757, saving model to weights-improvement-49-0.77.hdf5

Epoch 00050: val_acc did not improve from 0.76757 **----->** validation_split=0.1

acc: 70.73% **----> *Test set***

I have a problem to report the results of LSTM using AUC (two different score for area)



Compute Area Under the Receiver Operating Characteristic Curve (ROC AUC): 0.75679..

```
In [58]: M1=model.predict(X_test)   ##M1=predicted labels (binary)
         from sklearn.model_selection import cross_val_score
         from sklearn.metrics import roc_auc_score
         roc_auc_score(Y_test, M1)

Out[58]: 0.7567922897994202
```

```
In [60]: import matplotlib.pyplot as plt
         from sklearn.metrics import roc_curve, auc
         fpr = dict()
         tpr = dict()
         roc_auc = dict()
         for i in range(5):
             fpr[i], tpr[i], _ = roc_curve(Y_test[:, i],M1[:, i])
             roc_auc[i] = auc(fpr[i], tpr[i])

         # Compute micro-average ROC curve and ROC area
         fpr["micro"], tpr["micro"], _ = roc_curve(Y_test.ravel(), M1.ravel())
         roc_auc["micro"] = auc(fpr["micro"], tpr["micro"])
         plt.figure()
         lw = 2
         plt.plot(fpr[2], tpr[2], color='darkorange',
                  lw=lw, label='ROC curve (area = %0.2f)' % roc_auc[2])
         plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
         plt.xlim([0.0, 1.0])
         plt.ylim([0.0, 1.05])
         plt.xlabel('False Positive Rate')
         plt.ylabel('True Positive Rate')
         plt.title('Receiver operating characteristic example')
         plt.legend(loc="lower right")
         plt.show()
```