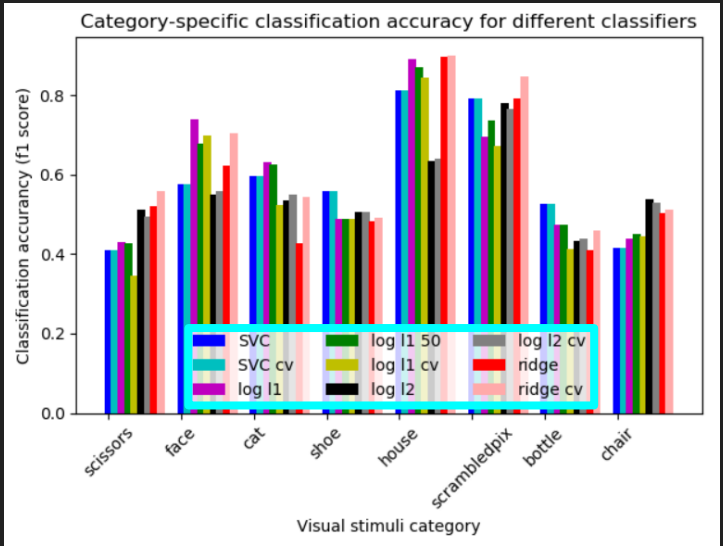
Categorical and binary classification in Visual object recognition task with dense neural network

JiHoon Kim

Nilearn:

Machine learning for Neuro-Imaging in Python

8.3.13. Different classifiers in decoding the Haxby dataset



Categorical classification

8.3.8. Decoding with ANOVA + SVM: face vs house in the Haxby dataset

8.3.8.6. Obtain prediction scores via cross validation

```
from sklearn.model_selection import LeaveOneGroupOut, cross_val_score

# Define the cross-validation scheme used for validation.
# Here we use a LeaveOneGroupOut cross-validation on the session group
# which corresponds to a leave-one-session-out
cv = LeaveOneGroupOut()

# Compute the prediction accuracy for the different folds (i.e. session)
cv_scores = cross_val_score(anova_svc, X, conditions, cv=cv, groups=session)

# Return the corresponding mean prediction accuracy
classification_accuracy = cv_scores.mean()

# Print the results
print("Classification accuracy: %.4f / Chance level: %f" %
      (classification_accuracy, 1. / len(conditions.unique())))
# Classification accuracy:  0.70370 / Chance level: 0.5000
```

Out:

```
Classification accuracy: 0.7037 / Chance level: 0.500000
```

Binary classification

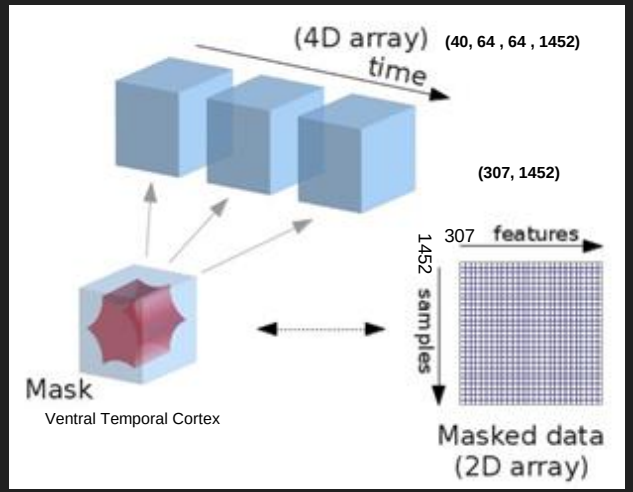
Data and methods

Haxby et al. (2001) dataset

8 objects stimuli labels

0 1 2 3 4 5 6 7

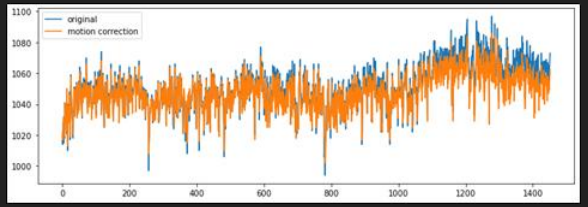
"It consists of 6 subjects with 12 runs per subject. In each run, the subjects passively viewed greyscale images of eight object categories"



https://nilearn.github.io/manipulating_images/manipulating_images.html

Preprocessing:

- Motion correction

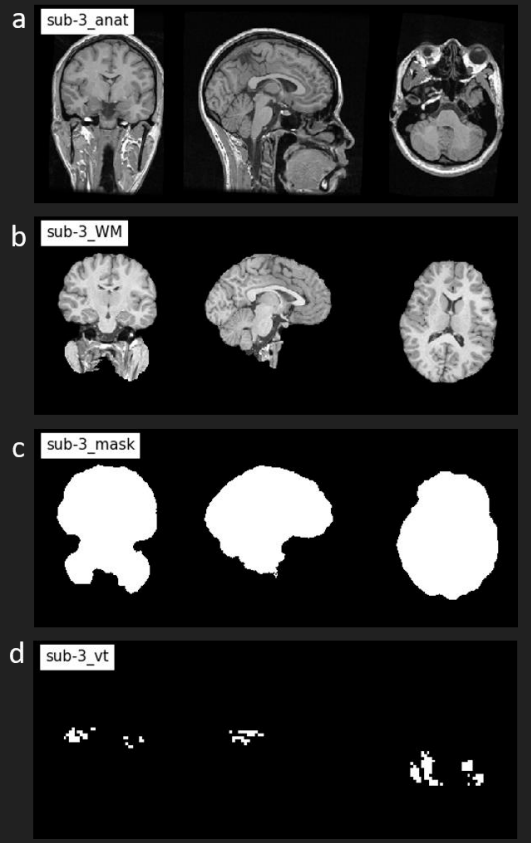


Subject 3 motion correction

- Smoothing fwhm = 16 mm
- Standardize the signal, z score
- Mask the Ventral Temporal cortex images

Data preparation for analysis:
Except subject 5, all subject contains 108 samples of eight object stimuli.

Subject 3, (307, 864)



Subject 3 mask process images

Dense neural network

A dense neural network with three dense layers and two dropout layers were used to classify the labels.

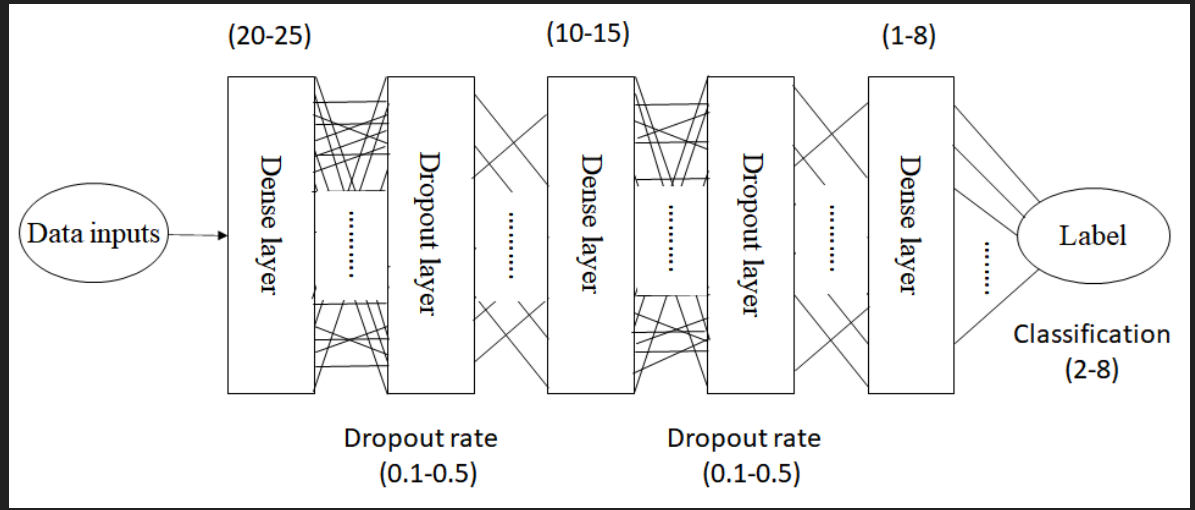
Grid search:
- Dense layer = 20 or 25
- Dense layer = 10 or 15
- Dropout rate = 0.1, 0.3, 0.5
- Epochs = 100, or 200
- Batch size = 25 or 50

K = 5 fold, Cross validation:
- Measure the Accuracy and loss

Dataset:
- Training set : Validation set : Test set

4 : 1 : 1

Schema of dense neural network



Categorical classification:
- Activation = relu
- Last dense layer = softmax
- Optimizer = Adam
- Loss = Categorical crossentropy

Layer (type)	Output Shape	Param #
dense_19 (Dense)	(None, 20)	6160
dropout_13 (Dropout)	(None, 20)	0
dense_20 (Dense)	(None, 15)	315
dropout_14 (Dropout)	(None, 15)	0
dense_21 (Dense)	(None, 8)	128

Total params: 6,603
Trainable params: 6,603
Non-trainable params: 0

Categorical classification model

Libraries and environment

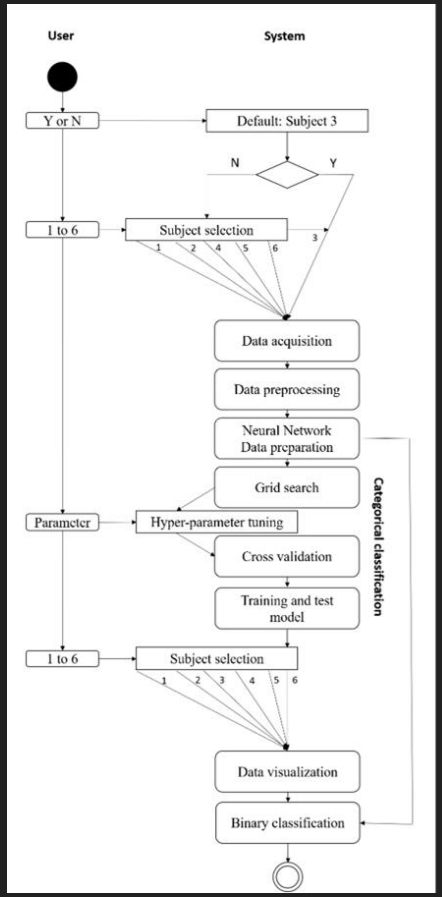
Environment	Version
Operating System	Window 10 Home
Virtual environment	Docker: jihoonkim2100/nnpp:init
Programming Language	Python 3.6.10
Libraries	Nibabel 3.1.1
	Nipype 1.6.0 dev0
	Nilearn 0.6.2
	Numpy 1.18.5
	Pandas 1.0.5
	Scikit-learn 0.23.1
	Tensoflow-cpu 1.13.1
	Keras 2.2.4
	Matplotlib 3.2.2
	Jupyter notebook 6.0.3

Implementation environment

```
import os
import sys
import time
import warnings

import numpy as np
import pandas as pd

import nibabel as nib
import nilearn.image as nimg
from nilearn.input_data import NiftiMasker
from nilearn.plotting import plot_anat
from nipype.interfaces import fsl
from nipype.interfaces.fsl import BET
from sklearn.model_selection import GridSearchCV

import keras
from keras import models
from keras import layers
from keras.models import Sequential
from keras.layers.core import Dense, Dropout, Activation
from keras.optimizers import Adam
from keras.utils import np_utils
from keras.utils import to_categorical
from keras.wrappers.scikit_learn import KerasClassifier

import matplotlib.pyplot as plt

warnings.filterwarnings("ignore")
print ( "--sys.version—")
print (sys . version)

# for reproducibility
np.random.seed(17072020)
```

libraries

Workflow and code



Workflow of analysis

Class	Function
DataAcquisition	**setdata** : initial setting **newfolder** : create a new folder **inputd** : load the data **outputd** : save the data **reorder** : reorder the data **acquisition** : do above functions
DataPreprocessing	**mask_image** : create a mask **mcflirt** : do motion correction **niftimakser** : preprocess smoothing, normalization, with VT mask.
NN	**categorical_label** : load the label **categorical_dataset** : data preparation **grid_search** : do grid search **gridsearch_model** : grid search model **cross_validation** : do cross validation **cross_validation_model** : CV model **test_model** : train and evaluate model **build_model** : Final model
Extra	**anat_mask** : show mask images **func_img** : show functional images **bianry_classification** : do binary classification on 'face' vs 'house'.
Interface	**preprocessing_interface** : get an input from user and proceed preprocessing **hyperparameter_interface** : receive inputs about the hyperparameter for CV and evaluate model from user **extra_image_interface** : generate images from input, subject number

Classes and functions

System design and implementation

```
##INSTRUCTIONS_STARTED############################################################
print('Welcome to neural network programming project interface')
func = Interface()

print('################_Analysis_started_#######################')
programming = time.time()

# Choose the default or other subject between 1 to 6
choice = input("Do you want to analysis Subject 3 or not? Type Y or N : ")

# Preprocess the feature and load the list
func_data,numbers_list,labels_list = func.preprocessing_interface(choice)

# Prepare the dataset
train_data,test_data,train_labels,test_labels,size = func.categorical_dataset(
    labels_list,func_data,numbers_list)

# Run the gridsearch
func.grid_search(train_data,train_labels)

# Run the cross validation k = 5
batch_size = input("Please type the batch size : ")
cv_parameter = func.hyperparameter_interface()
func.cross_validation(int(batch_size),cv_parameter,train_data,train_labels)

# Train and test the model
for final_epochs in [25,50,75,100,200]:
    func.test_model(final_epochs,cv_parameter)
print("total_computation_time :","%.2fs" %(time.time() - programming))
print('################_Analysis_completed_#######################')

##EXTRA_ANALYSIS:#################################################################
print('################_Analysis_extra_started###################')
programming2 = time.time()

# Choose the default or other subject between 1 to 6 based on previous analysis
choice = input("Did you test subject 3 or not? Type Y or N : ")

# Mask the image and shows the images: fMRI data signal, and mask iamge
func.extra_image_interface(choice)

# Do the binary classification and shows the result with model
func.binary_classification(func_data,labels_list,numbers_list)
print("total_computation_time :","%.2fs" %(time.time() - programming2))
print('################_Analysis_extra_completed#################')
##INSTRUCTIONS_COMPLETED##########################################################
```

Instructions

Result and discussion

Shell result on categorical classification on subject 3

Data acquisition to Preparation dataset

```
#################_Analysis_started_#####################################
Do you want to analysis Subject 3 or not? Type Y or N : Y
#################_Data_acquisition_started_#############################
ds000105/sub-3/func/bold.nii.gz loaded
Reorder ('R', 'A', 'S') completed
acquisition/sub-3_bold.nii.gz saved
acquisition_time : 18.40s
#################_Data_acquisition_completed_###########################
#################_Data_motion_correction_started_#######################
acquisition/sub-3_bold.nii.gz motion correction started
preprocessing/sub-3_bold.nii.gz motion correction completed
computation_time : 693.92s
#################_Data_motion_correction_completed_#####################
#################_Data_preprocessing_started_###########################
preprocessing/sub-3_bold.nii.gz
smoothing_fwhm : 12, and normalization started
preprocessing/sub-3_bold.nii.gz to func_data
smoothing_fwhm : 12, and normalization completed
computation_time : 20.59s
#################_Data_preprocessing_completed_#########################
#################_Categorical_label_checked_############################
labels_list: 864 loaded
computation_time : 0.03s
#################_Categorical_label_loaded_#############################
#################_Categorical_feature_checked_##########################
train_data, test_data : 720 144
Dataset standardization completed
size: 307  train_data: 720 and test_data: 144 loaded
computation_time : 0.01s
#################_Categorical_feature_loaded_###########################
```

Motion correction and Grid search computed more than 10 minutes.

Grid search to Model evaluation

```
#################_Grid_search_started_##################################
Best: 0.638889 using {'batch_size': 25, 'dropout_rate': 0.3, 'epochs': 100, 'neurons': 20, 'neurons2': 15}
computation_time : 799.65s
#################_Grid_search_completed_################################
Please type the batch size : 25
Please type the drop_out rate : 0.3
Please type the neuron layers 1 : 20
Please type the neuron layers 2 : 15
#################_Cross_validation_checked_#############################
current fold # 1
current fold # 2
current fold # 3
current fold # 4
current fold # 5
computation_time : 67.70s
#################_Cross_validation_completed_###########################
#################_Test_model_started_###################################

-- Multiclassification_model, epoch: 200 --
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_31 (Dense)             (None, 20)                6160
_________________________________________________________________
dropout_21 (Dropout)         (None, 20)                0
_________________________________________________________________
dense_32 (Dense)             (None, 15)                315
_________________________________________________________________
dropout_22 (Dropout)         (None, 15)                0
_________________________________________________________________
dense_33 (Dense)             (None, 8)                 128
=================================================================
Total params: 6,603
Trainable params: 6,603
Non-trainable params: 0
_________________________________________________________________

-- Evaluate --
144/144 [==============================] - 1s 4ms/step
loss : 0.8536845114496019
acc : 73.61%
model_epoch : 200, weight_and_architecture_saved
computation_time : 16.39s
#################_Test_model_completed_#################################
total_computation_time : 1689.07s
#################_Analysis_completed_###################################
```

Layer (type)	Output Shape	Param #
dense_19 (Dense)	(None, 20)	6160
dropout_13 (Dropout)	(None, 20)	0
dense_20 (Dense)	(None, 15)	315
dropout_14 (Dropout)	(None, 15)	0
dense_21 (Dense)	(None, 8)	128

Total params: 6,603
Trainable params: 6,603
Non-trainable params: 0

```
587     def test_model(self,data,data2):
588         def build_model(drop_out,neuron,neuron2):
589             model = models.Sequential()
590             model.add(layers.Dense(neuron,activation = 'relu',
591                                    input_shape = (size, )))
592             model.add(layers.Dropout(drop_out))
593             model.add(layers.Dense(neuron2,activation = 'relu'))
594             model.add(layers.Dropout(drop_out))
595             model.add(layers.Dense(8,activation = 'softmax'))
596             model.compile(optimizer = 'Adam',loss = 'categorical_crossentropy',
597                           metrics = ['acc'])
598             return model
599         # Model formulation
600         model = build_model(data2[0],data2[1],data2[2])
601         print(f"-- Multiclassification_model, epoch: {data} --")
602         model.summary()
603
604         # Model estimation
605         model.fit(train_data,train_labels,epochs=data,
606                   batch_size = int(batch_size), verbose = 0)
607         print("-- Evaluate --")
608
609         # Model evaluation
610         score = model.evaluate(test_data,test_labels,verbose = 1)
611         print('loss :',score[0])
612         print("%s : %.2f%%" %(model.metrics_names[1],score[1]*100))
613
614         # Save a model
615         model.save(f"model_epoch:{data}_weight.h5")
616         with open(f"model_epoch:{data}_architecture.json",'w') as f:
617             f.write(model.to_json())
618         print(f"model_epoch : {data}, weight_and_architecture_saved")
619         print("computation_time :","%.2fs" %(time.time() - preprocessing))
620         print('#################_Test_model_completed_#################')
```

Model

Categorical classification

Result and discussion

Table 1 – Best grid search on each subject. ACC; Accuracy, N1; First dense layer, N2; Second dense layer.

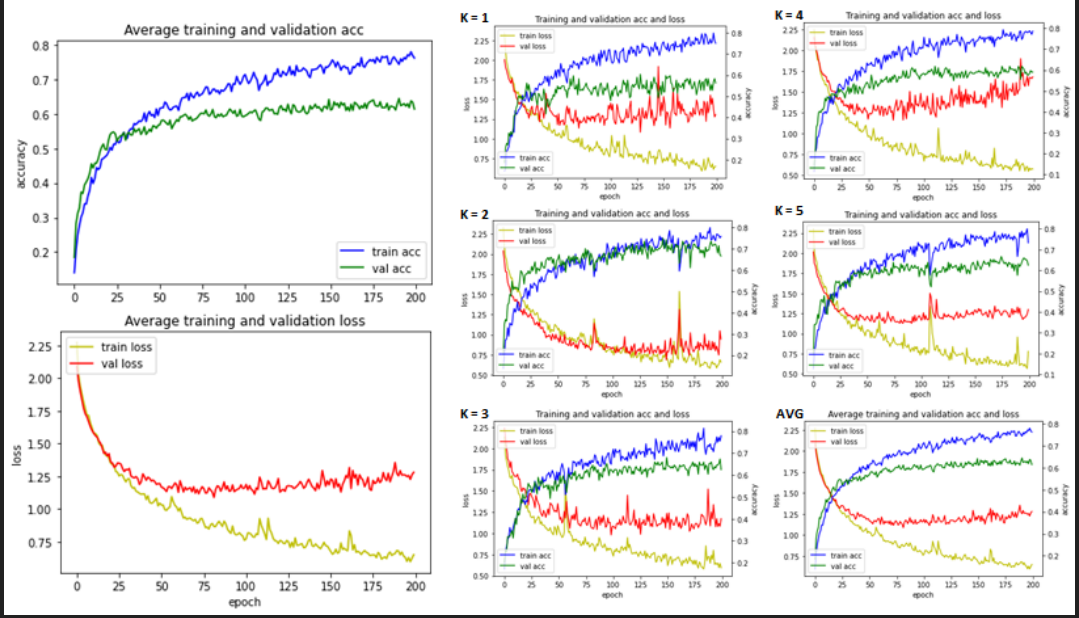
Subject	ACC	Batch size	Dropout rate	Epochs	N1	N2
1	0.611111	50	0.3	200	20	10
2	0.511111	50	0.3	100	20	15
3	0.638889	25	0.3	200	20	15
4	0.533333	50	0.1	200	20	10
5	0.654545	25	0.1	200	25	10
6	0.748611	25	0.1	200	20	10

Grid search result

Table 2 – Average accuracy of evaluation of test data.

Subject	Average accuracy	Highest accuracy, epochs	Lowest accuracy, epochs
1	62.64 %	63.19 % 25-75	61.11 %, 100
2	55.83 %	65.97 %, 200	40.97 %, 25
3	68.05 %	73.61 %, 200	55.56 %, 25
4	44.16 %	50 %, 200	39.58 %, 25
5	53.73 %	58.33 %, 200	51.25 %, 25
6	59.58 %	64.28 %, 75	52.08 %, 25

Categorical classification evaluation result



Cross validation result on subject 3

	Epochs					
Subject 3	**Average**	**25**	**50**	**75**	**100**	**200**
Accuracy (%)	68.05	55.56	70.83	69.44	70.83	73.61

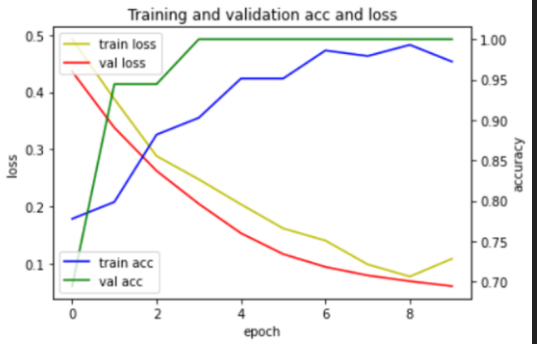
Model evaluation result on subject 3

Result and discussion

Binary classification

Layer (type)	Output Shape	Param #
dense_34 (Dense)	(None, 25)	7700
dropout_23 (Dropout)	(None, 25)	0
dense_35 (Dense)	(None, 15)	390
dropout_24 (Dropout)	(None, 15)	0
dense_36 (Dense)	(None, 1)	16

Total params: 8,106
Trainable params: 8,106
Non-trainable params: 0



-- Evaluate --
36/36 [==============================] - 0s 163us/step
loss : 0.09498310089111328
acc : 100.00%

Discussion

How to visualize the activation region on the neural network?

For group analysis, create common ROI mask needed

Using other neural network such as Convolutional neural network (CNN) model.

Open source

Data & Code

Haxby dataset

Docker hub

Github

OSF

Reference

- Haxby, J., Gobbini, M., Furey, M., Ishai, A., Schouten, J., and Pietrini, P. (2001). Distributed and overlapping representations of faces and objects in ventral temporal cortex. Science 293, 2425–2430.
- Chollet, F., & others. (2015). Keras. GitHub. Retrieved from https://github.com/fchollet/keras
- Abraham, A., Pedregosa, F., Eickenberg, M., Gervais, P., Mueller, A., Kossaifi, J., Gramfort, A., Thirion, B., & Varoquaux, G. (2014). Machine learning for neuroimaging with scikit-learn. Frontiers in neuroinformatics, 8, 14. https://doi.org/10.3389/fninf.2014.00014