Week 1: Genetic algorithms:

learning the basics and solve basic problem for one objective function.

Review:

- Optimization problems:

$$\min \quad f(x) = x + 10sin(2x)$$

Subject to

$$0 \leq x \leq 10$$

$$\text{Minimize:} \quad F = (X_1 - 1)^2 + (X_2 - 1)^2$$

$$\text{Subject to:} \quad X_2 \leq 0, \quad X_1 \geq 0$$

Week 2: Genetic algorithms for two objective functions:

learning the basics and solve basic problem for two objective functions.

Week 3: Deep learning:

learning the basics

Week 4: Deep learning: Creating CNN for a data set

(share the parameters they play with, share a data set, define the objective function: validation accuracy), colab example. Cat and dog,

https://www.youtube.com/watch?v=j-3vuBynnOE&list=PLQVvvaa0QuDfhTox0AjmQ6tvTgMBZBEXN&index=2

using first 2000 cat, and first 2000 dogs' images. Create a data set. Split it into validation, and test.

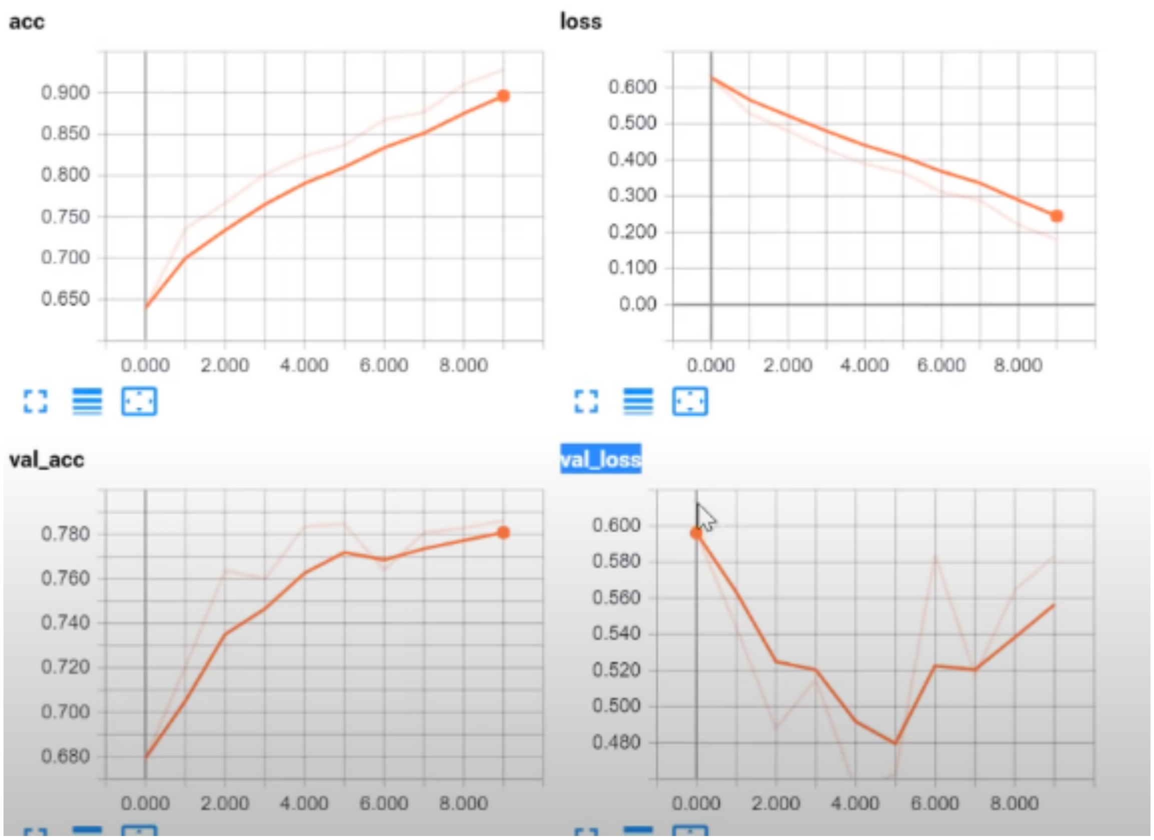
```
import pandas as pd

import tensorflow as tf

from sklearn.preprocessing import RobustScaler,MinMaxScaler,StandardScaler

from sklearn.model_selection import GridSearchCV,train_test_split,cross_va
l_score
```

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.impute import SimpleImputer
from sklearn.svm import SVC
from sklearn import metrics
from sklearn.metrics import r2_score
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
import numpy as np
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LinearRegression
from mpl_toolkits import mplot3d
from scipy import stats
from sklearn.kernel_ridge import KernelRidge
from sklearn.ensemble import RandomForestRegressor
from keras.layers import Dense
from keras.models import Sequential
#from keras.wrappers.scikit_learn import KerasRegressor
from keras.wrappers.scikit_learn import KerasClassifier
from keras.callbacks import EarlyStopping
from keras import callbacks
import seaborn as sns
from keras.utils import np_utils
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, LSTM
```

```
X_train=…
Y_train=…
X = X/255.0
```

```
# creating 3 data sets : training, validation, test.

X_train_main,X_test,Y_train_main,Y_test=train_test_split(X_train,Y_train,test_size=0.1,random_state=42)

X_train_main,X_val,Y_train_main,Y_val=train_test_split(X_train_main,Y_train_main,test_size=0.1,random_state=42)



import keras
from keras.utils import np_utils

from keras.models import model_from_json

from keras.models import Sequential

from keras.layers import Convolution2D, MaxPooling2D, Dense, Dropout, Activation, Flatten

from sklearn.model_selection import KFold

import tensorflow as tf

from tensorflow.keras.callbacks import EarlyStopping

from tensorflow.keras import callbacks


print('creating model')

# CNN, MAX, CNN, MAX, CNN, MAX, CNN, MAX, CNN, AVE, FULLY



model = Sequential()

model.add(Conv2D(256, (3, 3), input_shape=X.shape[1:]))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(256, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))


.

.
```

```
.
model.add(Flatten())  # this converts our 3D feature maps to 1D feature vectors

model.add(Dense(64))

model.add(Dense(1))
model.add(Activation('sigmoid'))


model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])


#Model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])

Model.summary()

opt = keras.optimizers.Adam()

Model.compile(loss='categorical_crossentropy', optimizer=opt,metrics=[tf.keras.metrics.Recall()])

  # or use validation split to split the the data

  # Early Stopping, which takes the validation set error into consideration to prevent overfitting

monitor=EarlyStopping(monitor='val_recall', mode='max', verbose=1, patience=2)

checkpointer=callbacks.ModelCheckpoint(filepath="3 Azure models/2 portion 200/1 one eigen/classification.hdf5",verbose=1,save_best_only=True)

history_train1=Model.fit(X_train_main,Y_train_main,epochs=100,batch_size=500 ,callbacks=[checkpointer],validation_data=(X_val,Y_val), verbose=1)
```




```
loss, accuracy= model.evaluate(X_test, Y_test, verbose=0)
```

Week 5: Integration: GA + DL integration

Goals: define objective function, choose the variables, code development

- deep learning codes

First results of your senior project reports will be produced this week.

Review:

- Optimization problems:

$$\min \quad f(x) = x + 10sin(2x)$$

Subject to

$$0 \leq x \leq 10$$

Minimize: $F = (X_1 - 1)^2 + (X_2 - 1)^2$

Subject to:

$$X_2 \leq 0$$
$$X_1 \geq 0$$

- What is the optimization problem we are trying to solve?

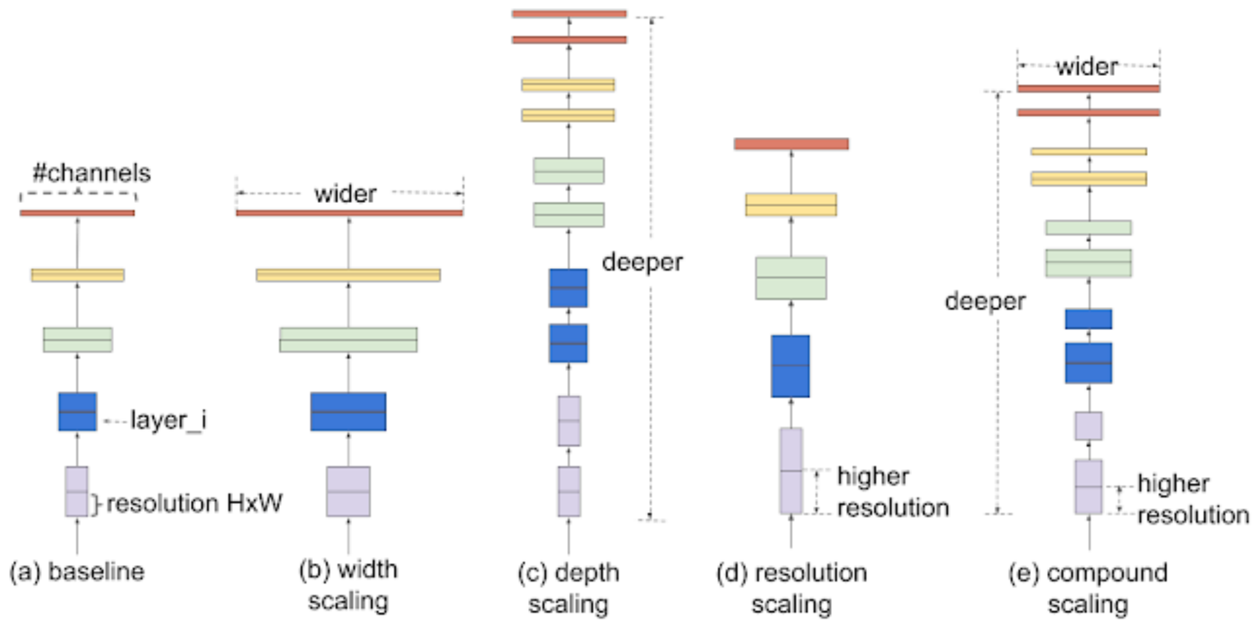
Minimize: Validation_LOSS(X_1)

Subjected to $1<X_1<80$ (number of filters in first convolutional layer)

….

The variables all contribute to the response of the CNN. This optimization problem is subjected to several constraints

We usually find the values of CNN parameters such as filter number by trial and error.



we don't want to use grid search.

hyper parameters (week 5)

- There are many hyper parameters in CNN that we can play with. The chosen hyper parameters for week 5.

convolutional layer: filter size, number of filters

max pooling: filter size.

Dense layer: number of nodes.

Batch size: 32, 64, 128, 256 (just 4 values).

- My suggestion: We can start with just one variable "X_1 =number of filters", if successful considers more variables.

- Additional hyper parameters (If you have time add them to your model, otherwise ignore them for now).

Dropout

activation function: relu, tanh, ...

optimizer type: adam, nadam, ...

Learning rate: 0.01, 0.001, ….

Training Hyperparameters	Spatial Feature Learning Hyperparameters
Learning rate, ϵ	Patch size, m
Momentum, α	Convolutional layers, g
Learning rate decay, ϵ_d	Fully connected layers, t
Early stopping patience,	Number of filters, k
Maximum number of epochs, e_n	Filter size, h
Weight decay, λ	
Dropout rate, d_r	

Total number of layers (week 6)

Total number of pair layers (conv + max pooling) before two last dense layers. (CP)

Total number of pair layers (conv + conv+ max pooling) before two last dense layers. (CCP)

E.g.

CPDD

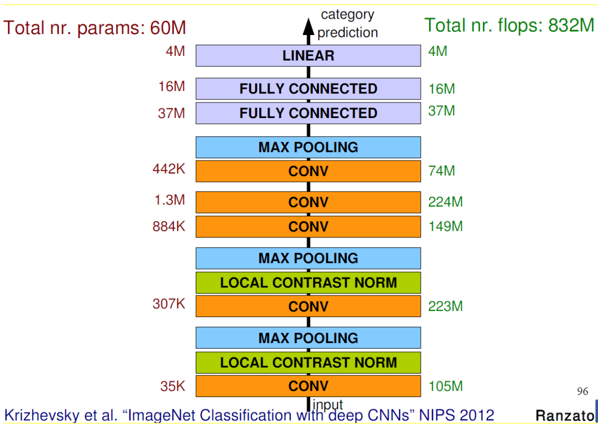
CPCPDD

CCPCPDD

…

order in the layers: e.g. conv, pooling,

Week 6: Integration: Adding more variables



Total number of pair layers (conv + max pooling) before two last dense layers. (CP)

Total number of pair layers (conv + conv+ max pooling) before two last dense layers. (CCP)

E.g.

CPDD

CPCPDD

CCPCPDD

…

order in the layers: e.g. conv, pooling,

Week 8: Testing:
Vermeer data set

Week 9: Adding more complexity / Possible publication:
Try to enhance the technique and compare the performance with earlier works.

Week 10: Adding more complexity / Possible publication:
Try to enhance the technique and compare the performance with earlier works.

Week 11: Future plan:
Writing a paper, Vermeer opportunities