**What is Keras ?**

👉Keras is an open source neural network library written in Python that can run smoothly on the CPU and GPU. Today, I’m going to use Tensorflow in background. However available like Theano and CNTK backend engines. You can use whatever you want. Now, let’s import the libraries.

import numpy as np  
import pandas as pd   
from keras.preprocessing.image import ImageDataGenerator, load\_img  
from keras.utils import to\_categorical  
from sklearn.model\_selection import train\_test\_split  
import matplotlib.pyplot as plt  
import random  
import os

👉 In this way we emphasized required libraries for education. When we run the code, as it is seen feedback have to return.



Expected feedback

👉 Then, in working directory , we can list to see train and test data that will use with os.listdir( ). We can see with this command consist of which data in directory.

**Examination of Data Set**

You can download for free from [**Dogs and Cats**](https://www.kaggle.com/c/dogs-vs-cats) dataset in Kaggle.

print(os.listdir("../Data/"))

print(os.listdir("../Data/"))



Folders in Directory

By the way, this project does classification using deep learning of multi layer categories images. We’re going to use as of dog and cat labels in images.

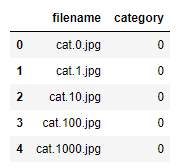
filenames = os.listdir(“Data/train”)  
categories = []  
for filename in filenames:  
 category = filename.split(‘.’)[0]  
 if category == ‘dog’:  
 categories.append(1)  
 else:  
 categories.append(0)

👉It changed as 1 or 0 labels, dogs and cats labels in train data. The labels of dogs and cats has changed as 1 or 0.

df = pd.DataFrame({  
 'filename': filenames,  
 'category': categories  
})

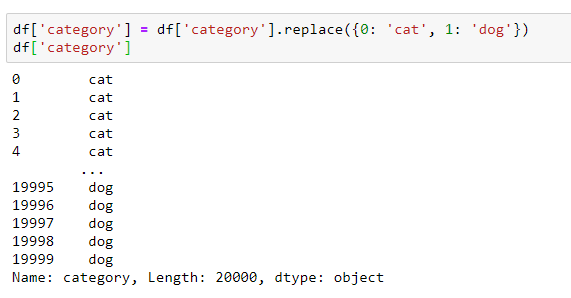
👉We can read and write data with Pandas library .We can associate to file name and category as filenames and categories in DataFrame. The category variable specifies whether an image is a cat or a dog. Afterward first 5 data was retrieval with head( ) command as default in this DataFrame.

df.head( )



First 5 data retrieval

👉Firstly, cats and dogs for processing converted to 0 or 1.



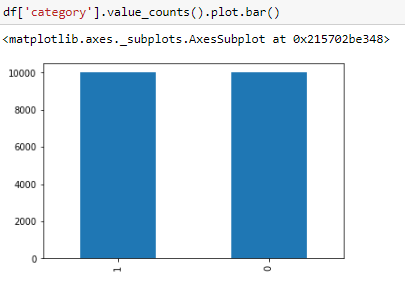
Control of 20000 images

test\_df[‘category’]=test\_df[‘category’].replace({‘dog’: 1, ‘cat’: 0 })

👉 For later use as the latest operation, 1 and 0 labels were reconverted to cat and dog categories in the testing process.

df[‘category’].value\_counts().plot.bar()

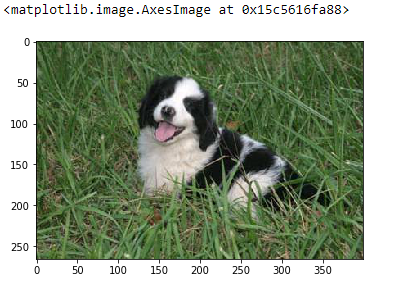
👉By counting the data with the 0 and 1 labels in the category class in the data set above with value\_counts (), the bar graph is extracted with the bar () function according to the matplotlib.



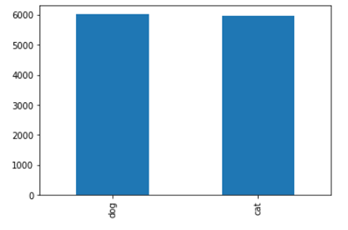
As predict 10000 cats and dogs images

**Retrieve Random Image from Data Set**

sample = random.choice(filenames)  
image = load\_img(“Data/train/”+sample)  
plt.imshow(image)



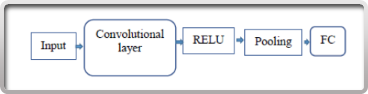
If we perform the separation of the data set on a total of 20000 images, there are 12000 train and 8000 test data. 60% of the data set in this project is devoted to education and 40% to test data set.



Graphicalization of 12000 training set values

❗️CNN is a neural network model used to classify the image by processing it with various layers. The presence of the object in that area is classified by CNN with different areas of interest from the image.

❗️For we will use a CNN neural network model, it is necessary to know the layers that the structure provides to us and to design the model well. I will show the neural network architecture to be used in this project. The CNN model consists of Convolutional Layer, Pooling Layer and Fully Connected. Generally, excessive learning is prevented by various regularization methods (Dropout, Early Stopping, etc.) for the prevention of overfitting. I did not use any regularization method in this study, you can use it to get better results.



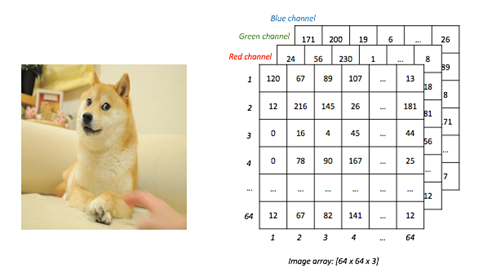
CNN Layer Architecture

👉 The project uses a convolutional layer, RELU function as activation function, a convolutional neural network consisting of a pooling layer and a fully connected layer.

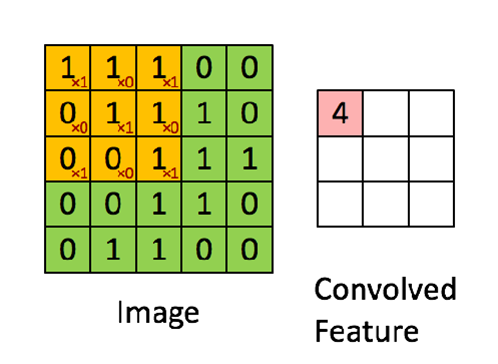
**STRUCTURE of LAYER MODEL**

For classification purposes, it has an architecture such that the convolutional network is [INPUT-CONV-RELU-POOL-FC]. The first structure, INPUT, contains the image data to be used. 20000 cat and dog pictures in the data set were selected as the input images. CNN neural network model was selected to classify these images.

👉First, Convolution layer called Convolution has been created with a certain number of kernels (kernel\_size can be selected arbitrarily). With this convolution kernel, filters are applied to extract properties on the image. This layer is the building block of the CNN model. You can think of it as pixel-by-pixel application like of Mean, Gaussian, Laplas, Sobel filters on image processing. My image size is 64 x 64 and IMAGE\_CHANNEL is 3. The following shows how this color channel, selected as RGB, behaves on image pixels.

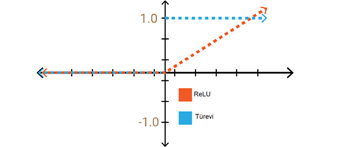


Applying Convolution Filter with RGB Channels



Feature Extraction with Convolution Layer

🤗Through this Convolution layer, feature extraction, called feature detection was performed on the images. Examples of these features are the cat’s eye, the dog’s tail. In this way, the images are deepened. Then the activation function RELU was applied on the feature map.



ReLU Function

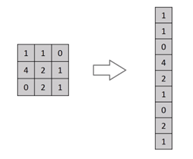
👉Activation in multilayer neural networks is intensive, meaning that it requires a lot of processing. Some neurons in the network are active, and activation is sparse, so it is desirable to get an efficient computing load. With ReLU, the calculation load is more efficient than other activation functions.

👉On the images and properties coming from the convolution layer, the Pooling layer, called Pooling, has a say. With this layer, the complexity of the model is reduced by reducing the number of parameters without degrading performance. It will make every recognition in the image to avoid affecting the classification. At the same time, this process also prevents over-learning process called Overfitting. MaxPooling is used as pooling structure. In this way, the maximum number of kernel sizes is taken. With this process, smaller outputs containing sufficient information are used for the correct decision of the neural network.



Visualizing of MaxPooling

👉For the last layer of architecture and the **Fully Connected** layer, the data will be converted into a single input matrix with getting kernels. The neural networks in the Fully Connected Layer are fully interconnected. Each neuron has a separate weight value. The network to be created here will consist only of such layers. This is done by **Flatten ( )** is done through. This data is then reconciled with **Dense** by a density filter. In this layer, the input data received at the input is prepared for use in Fully Connected Layer.



Getting single input data

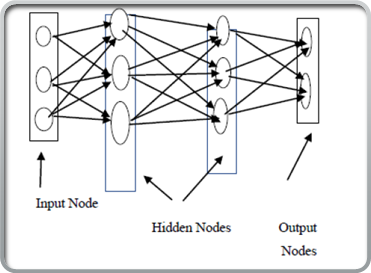


Adding of Dense Layer

**Creating CNN Architecture in Python Code**

from keras.models import Sequential  
from keras.layers import Conv2D, MaxPooling2D, Dropout, Flatten, Dense, Activation, BatchNormalizationmodel model = Sequential()model.add(Conv2D(64, (3, 3), activation='relu',use\_bias=True, input\_shape=(IMAGE\_WIDTH, IMAGE\_HEIGHT, IMAGE\_CHANNELS)))  
model.add(MaxPooling2D(pool\_size=(2, 2)))model.add(Flatten())  
model.add(Dense(512, activation = 'relu',use\_bias=True))  
model.add(Dense(2, activation = 'relu',use\_bias=True))model.compile(loss='categorical\_crossentropy', optimizer='rmsprop', metrics=['accuracy'])  
model.summary()

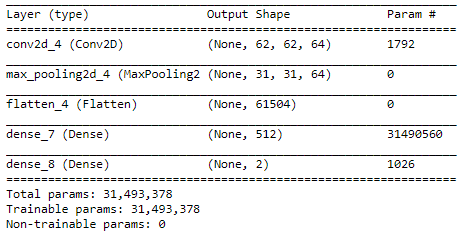
👉In the Convolution layer, I have 64 filters and kernel\_size (3,3). In the first Conv layer, the width and height values of the image must be entered. Filtering will be performed according to these values. The activation function in it has already been specified as relu. If you wish, you can use softmax in the Dense layer. Bias values must also be included in the formula in Z signal calculated during weight updates in Artificial Neural Networks. Because bias values are also used in this article, I specified use\_bias = True. And on the pooling layer, I created a pool\_size (2.2) and reduced the data. Since there were 2 hidden layers in my neural network construction, I made the transaction according to Fully Connected.



Structure of ANN

❗️As you know, there is a cost calculation in Artificial Neural Networks. We want the cost to be minimum in this calculation. Therefore, it is very important for us to calculate the loss value.

👉In order to compile the model we created, the loss parameter to be used must be selected. For this, there are very good explanations about the [**use of loss**](https://keras.io/losses/) in Keras. I used **categorical\_crossentropy** because I made a two-category classification.



Summary of Model

👉The batch\_size variable to be used in the project has been equalized to 100.

batch\_size=100

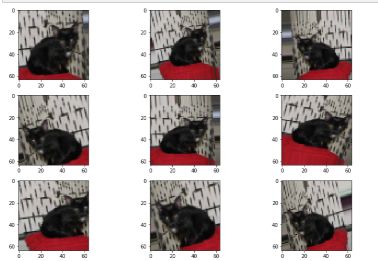
👉Then, to increase the data, we switched to Data Augmentation process. Data increment method such as zooming, zooming, rotating, horizontal turning was applied by specifying cutting and rotation intervals.

train\_datagen = ImageDataGenerator(  
 rotation\_range=15,  
 rescale=1./255,  
 shear\_range=0.1,  
 zoom\_range=0.2,  
 horizontal\_flip=True,  
 width\_shift\_range=0.1,  
 height\_shift\_range=0.1  
)#Data augmentation  
train\_generator = train\_datagen.flow\_from\_dataframe(  
 train\_df,   
 "Data/train/",  
 x\_col='filename',  
 y\_col='category',  
 target\_size=IMAGE\_SIZE,  
 class\_mode='categorical',  
 batch\_size=batch\_size  
)



Validated train set for 2 classes

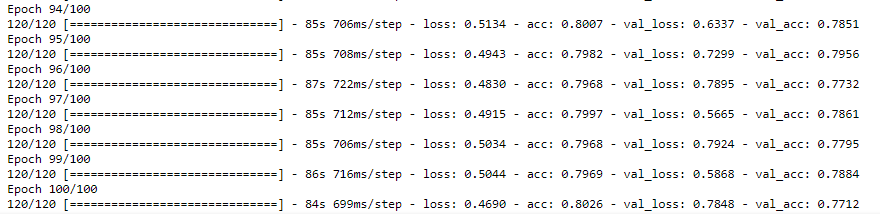
plt.figure(figsize=(12, 12))  
for i in range(0, 9):  
 plt.subplot(5, 3, i+1)  
 for X\_batch, Y\_batch in example\_generator:  
 image = X\_batch[0]  
 plt.imshow(image)  
 break  
plt.tight\_layout()  
plt.show()



Graphicalization of data augmentation results

👉Subsequently, 100, 500 and 1000 epochs were used to train the data, respectively. After these values, the accuracy value started to decrease due to the loss value. A process of 100 iterations is shown below.

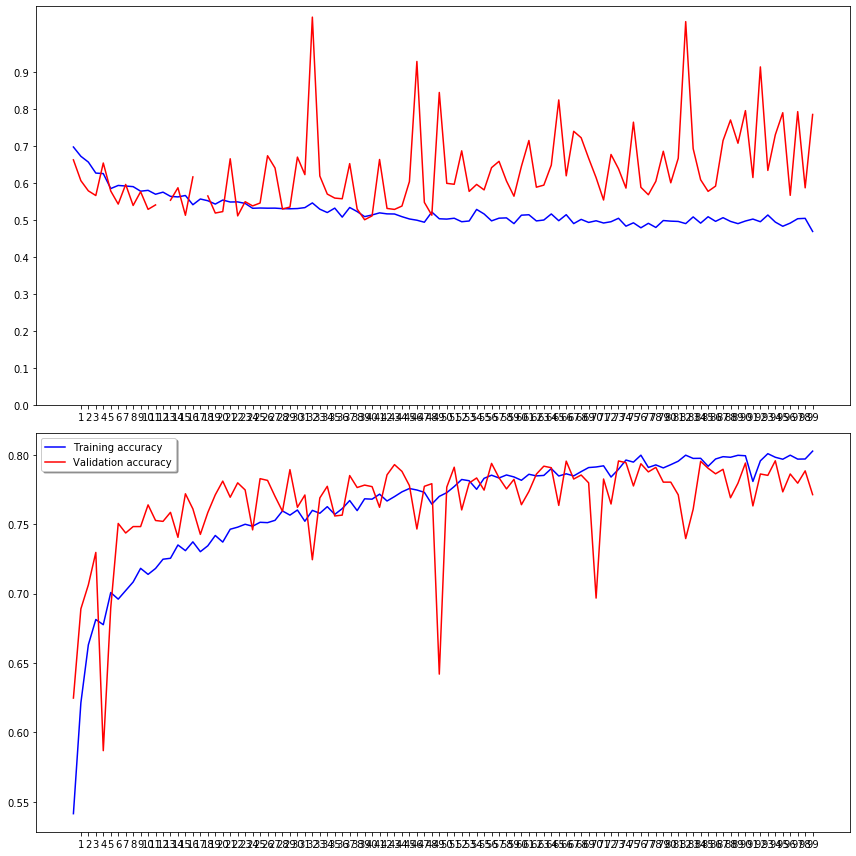
epochs=1 if FAST\_RUN else 100   
history = model.fit\_generator(  
 train\_generator,   
 epochs=epochs,  
 validation\_data=validation\_generator,  
 validation\_steps=total\_validate//batch\_size,  
 steps\_per\_epoch=total\_train//batch\_size,  
)



Loss and Accuracy Results for 100 Epoch

👉For the change of loss values and accuracy values graph, graphs were created by running the following code piece.

fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(12, 12))  
ax1.plot(history.history['loss'], color='b', label="Training loss")  
ax1.plot(history.history['val\_loss'], color='r', label="validation loss")  
ax1.set\_xticks(np.arange(1, epochs, 1))  
ax1.set\_yticks(np.arange(0, 1, 0.1))ax2.plot(history.history['acc'], color='b', label="Training accuracy")  
ax2.plot(history.history['val\_acc'], color='r',label="Validation accuracy")  
ax2.set\_xticks(np.arange(1, epochs, 1))legend = plt.legend(loc='best', shadow=True)  
plt.tight\_layout()  
plt.show()



Loss & Accuracy Change Chart

👉 Then, the estimation phase was passed and the test images in the folder were submitted to the machine.

test\_filenames = os.listdir("Data/test1")  
test\_df = pd.DataFrame({  
 'filename': test\_filenames  
})  
nb\_samples = test\_df.shape[0]

❗️ Also data augmentation is performed on test images.

test\_gen = ImageDataGenerator(rescale=1./255)  
test\_generator = test\_gen.flow\_from\_dataframe(  
 test\_df,   
 "Data/test1/",   
 x\_col='filename',  
 y\_col=None,  
 class\_mode=None,  
 target\_size=IMAGE\_SIZE,  
 batch\_size=batch\_size,  
 shuffle=False  
)



👉The model was predicted by the predict method, and the label created in the above category represents 1: dog 0: cat, but this labeling has now been modified below.

predict = model.predict\_generator(test\_generator, steps=np.ceil(nb\_samples/batch\_size))test\_df['category'] = np.argmax(predict, axis=-1)  
label\_map = dict((v,k) for k,v in train\_generator.class\_indices.items())  
test\_df['category'] = test\_df['category'].replace(label\_map)  
test\_df['category'] = test\_df['category'].replace({ 'dog': 1, 'cat': 0 })

👉 Then, the prediction result was printed by labeling the CSV file.

submission\_df = test\_df.copy()  
submission\_df[‘id’] =submission\_df[‘filename’].str.split(‘.’).str[0]  
submission\_df[‘label’] =submission\_df[‘category’]  
submission\_df.drop([‘filename’, ‘category’], axis=1, inplace=True)  
submission\_df.to\_csv(‘Test\_submission.csv’, index=False)



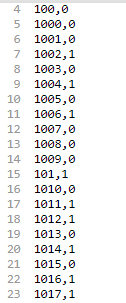
The resulting CSV file

To ensure control of the generated CSV file, first consider the image with ids 1085 and 1079. The image shows the cat. In the CSV file, the categories are specified as 0 and 1. This information is also provided in the folder by controlling it.



Control of prediction

👉View of the CSV file containing the test data;



Labels in CSV file

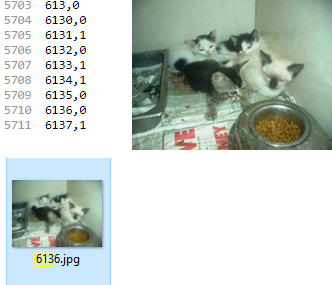


Image 6136: cat