COMP-SCI 55510 - Advanced Software Engineering

Project Presentation (May 7th, 2019)

Dhabbah,Khalid Mohammed A(6)
Doss,Corey Jason(8)
Sun,Chen (23)
Xie,Tian Cheng(25)

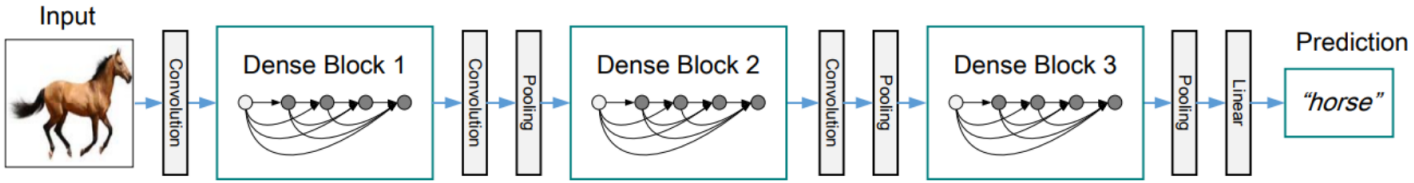
Model as a service

Our platform is an online platform that uses machine learning (ML) and deep learning (DL) models. The platform offers seven different models that provide users with information about uploading their own images and identifying images. Users can choose the most suitable model.

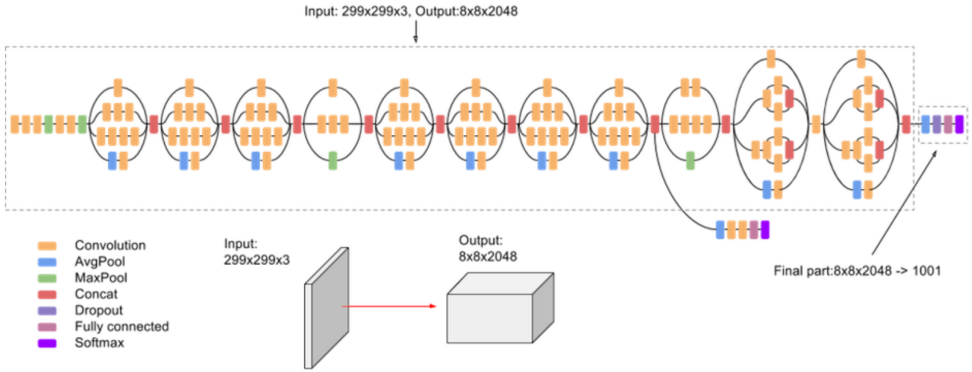
7 Models

- DenseNet121
- InceptionV3
- MobileNet
- Nasnet
- ResNet50
- VGG16
- xception

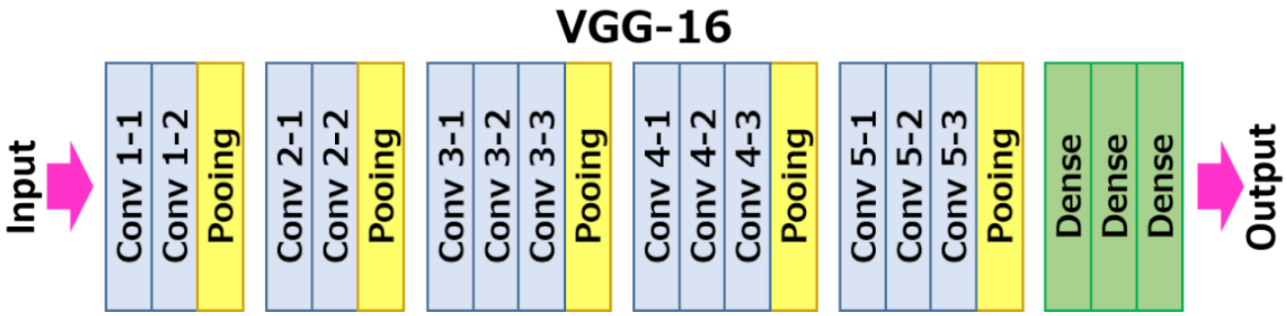
- DenseNet121:It is a logical extension of ResNet.DenseNet connects the output of the previous layer instead of using s



- InceptionV3 :Inception v3 is a widely-used image recognition model that has been shown to attain greater than 78.1% accuracy on the ImageNet dataset.



- ResNet50：It is a deep convolutional networks for classification.
- VGG16：It convolutional neural network is a model proposed by Oxford University in 2014.It is simple and practical, the most popular of which is VGG-16, which is a 16-layer model.

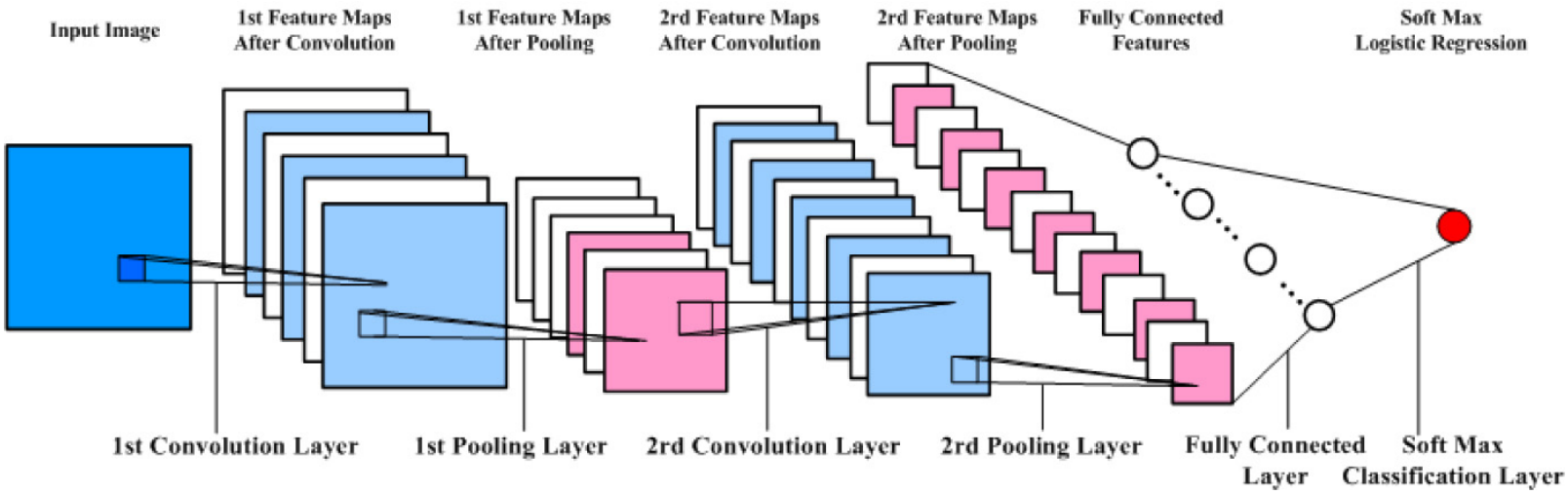


- MobileNet:It is an efficient model for mobile and embedded devices. MobileNets is based on streamlined, using depth wise separable convolutions to build lightweight deep neural networks.
- Nasnet :It classifies images. Given an image, the NasNet network will output probabilities of the different classes that an image could potentially belong to.
- xception:The mapping of cross-channel correlation and spatial correlation in the feature map of the convolutional neural network can be completely decoupled. This assumption is an extreme assumption in the Inception structure.

Models training

- Convolutional Neural Network should be involved in order to do some significant steps, which are embedding, convolution, pooling, flattening, full connection.

- Also, loss and accuracy are playing a significant role. The loss is the number of errors in prediction, so it should be decreased, and the accuracy should be increased.



Here is how to get the output from a pre-trained CNN model we received using Jupyter Notebook.

Loading the model

1. First, Loading the model:

```
#Tensorflow and leras should be imported
import tensorflow as tf
from keras.models import load_model
model = load_model('CNN011019-223859_model.h5')
```

2. Second, many functions can be used getting information about the model, such as model.summary(): to get all the layers.

2. Third, save a json file for this model for future developments.

```
# save as JSON
json_string = model.to_json()
with open('CNN011019-223859_model.json', 'w') as file:
file.write(json_string)
```

4. Fourth, compiling the model

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

Loading the model(Cont..)

5. Fifth, passing a picture through the layers

```
#pip3 install opencv-contrib-python --user
import cv2
import numpy as np
img = cv2.imread('1.jpg')
print(img.shape)
img = cv2.resize(img,(64,64))
print(img.shape)
img = np.reshape(img,[1,64,64,3])
```

6. Finally, Getting the output

```
classes = model.predict_classes(img)
print(*classes)
```

How the backend works

```python
@app.route('/predict', methods=['POST'])
def upload():
    if request.method == 'POST':
        # Get the file from post request
        f = request.files['file']

        # Save the file to ./uploads
        basepath = os.path.dirname(__file__)
        file_path = os.path.join(
            basepath, 'uploads', secure_filename(f.filename))
        f.save(file_path)

        # Make prediction
        preds = model_predict(file_path, model)

        pred_class = decode_predictions(preds, top=5)    # ImageNet Decode
        result = str(pred_class[0][0][1])                # Convert to string
        return 'The model DenseNet121 predicts this image as : ' + result
    return None
```

the frontend will use the RESTful API from the backend

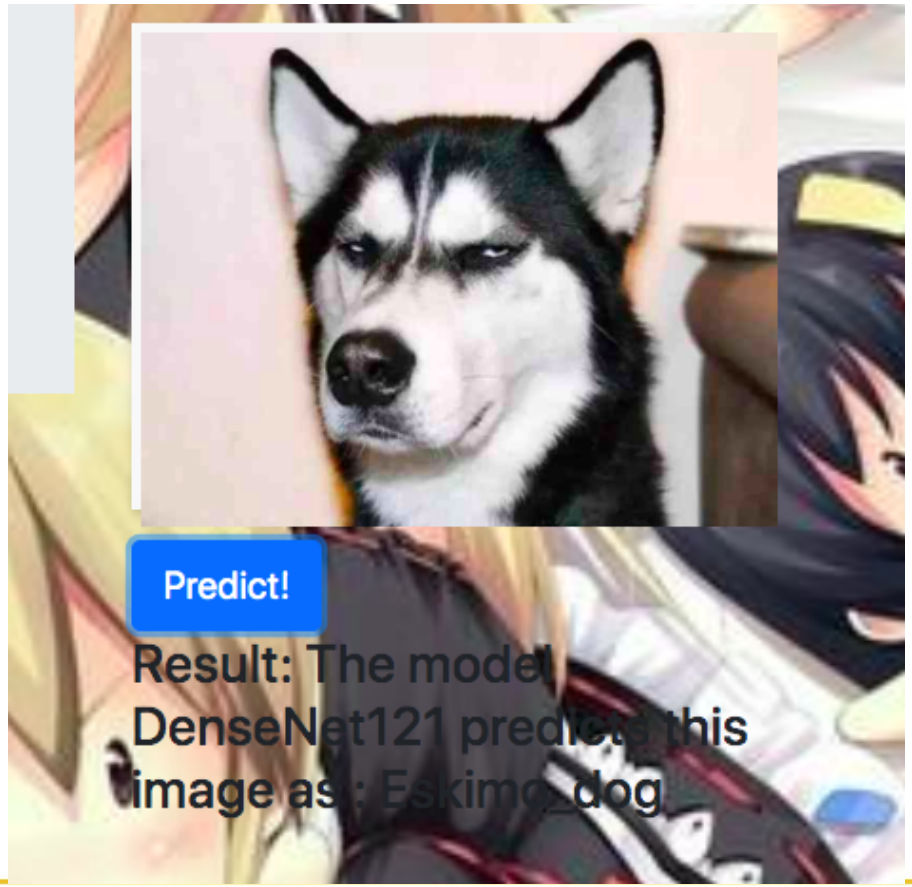
How the backend works

```
model = DenseNet121(weights='imagenet')

def model_predict(img_path, model):
    img = image.load_img(img_path, target_size=(224, 224))
    x = image.img_to_array(img) # Preprocessing the image
    x = np.expand_dims(x, axis=0) # x = np.true_divide(x, 255)

    preds = model.predict(preprocess_input(x))
    return preds
```

We are using DenseNer121 model here, and the function job is to resize the image file and transfer it to array and divide it in order to pass it through all the layers.

Deployment

- Get the prediction result by DenseNet121



Reference

1. https://arxiv.org/abs/1704.04861
2. https://www.modeldepot.io/jbrandowski/nasnet-mobile
3. https://neurohive.io/en/popular-networks/vgg16/
4. https://www.kaggle.com/lamhoangtung/densenet-121-lb-0-925
5. https://www.kaggle.com/pytorch/densenet121
6. https://www.jianshu.com/p/cc830a6ed54b
7. https://www.osapublishing.org/boe/fulltext.cfm?uri=boe-8-5-2732&id=363511

Thank you!