NYU

Object Detection Based on FCN

Chen shen and Yaochen Liu
5/16/2018

NYU

Introduction

Object Detection and Segmentation

Object Detection and Segmentation

- One of the most popular area of research in the field of Computer Vision
- Can date back to 1960s when researchers were interested in Object Recognition and Classification
- Limited by the temporal computing power so good results were rare

The assistance of Machine Learning

- Since the rapid development of Machine Learning, especially the Convolutional Neuron Network (Convnet), many researchers managed to apply their knowledge in Convnet to the realm of Object Detection
- Performance of detection is improved

Detection & Segmentation in Conventional Convnet

- Even involving Convnet into object detection and segmentation has obtained eye-catching achievements
- However, in conventional Convnet, the surrounding pixels are always taken into consideration when it comes to classify one specific pixel
- Some drawbacks are inevitably brought about from the mechanism

Drawbacks of Conventional Convnet

- Storage spending are relatively high when attached pixels' information are considered
- High probability that attached pixels are identical, which makes overlapped and futile computation
- Performance is limited by the dimension of weights and pixels

NYU

Considering the cons in conventional Convnet, many improved methods are invented and developed, one of which is Fully Convolutional Network (FCN).

NYU

FCN

FCN utilized in Segmentation

The history of FCN in Detection & Segmentation

- To our knowledge, firstly introduced by Matan et al. [1]. In their work, classic LeNet [2] is extended to recognize one-dimensional strings of digits
- Due to limitation of one-dimension, Wolf and Platt [3] has enlarged to two-dimension when handling the problem of detection scores for the four corners of postal address blocks
- Since then, many improvements and enhancements are conducted worldwide, such as
    - Convnet for coarse multiclass segmentation based on FCN by Ning et al. [4].
    - Boundary Prediction for electron microscopy by Ciresan et al. [5].
    - Hybrid Convnet model for natural image by Ganin et al. [6].

One remarkable work

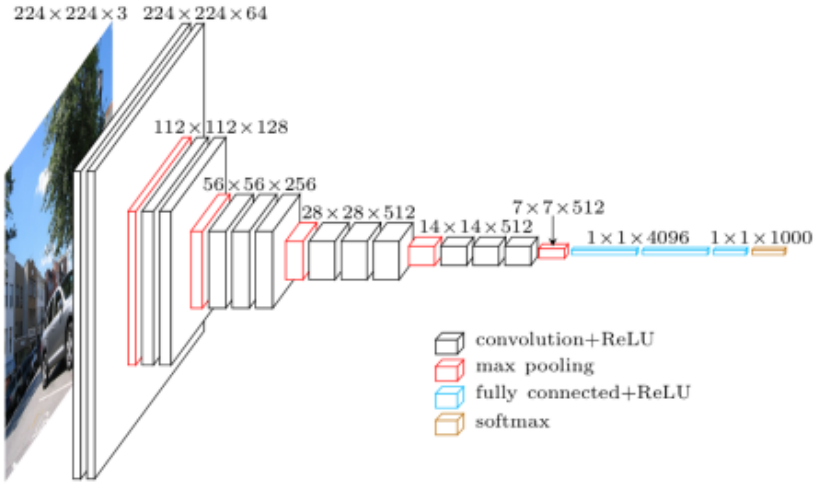
- The method of “shift-and-stitch” dense output and in-network upsampling proposed by Shelhamer et al. [7] has aroused worldwide attentions and obtained remarkable achievements.

Features and State-of-arts in the work

- In their method, deep classification architectures are adapted and extended to use image classification as supervised per-training.
- In addition, the extended architectures are exploiting fine-tune Fully Convolutional to learn simply and efficiently from whole image inputs and ground truths.

Basic Knowledge of FCN models

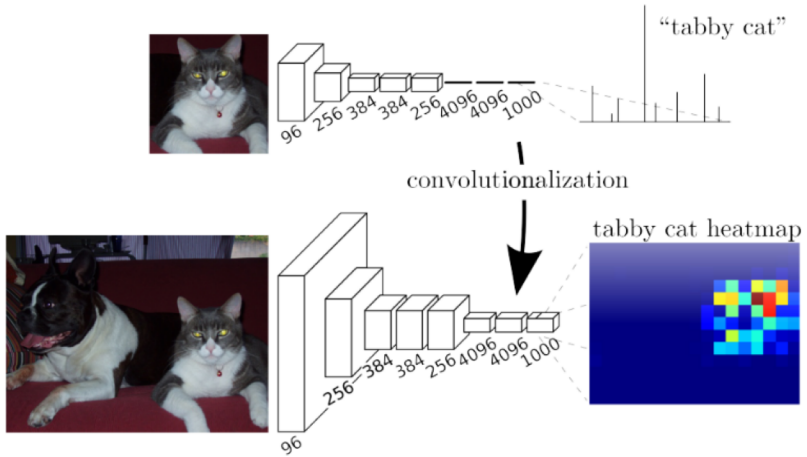
- **The conventional Convnet**: the last few layers of a given model are fully connected layers with activation function and an output layer of Softmax function, which transform the pixels in a given image into feature net for classification.



Resource:https://blog.heuritech.com/2016/02/29/a-brief-report-of-the-heuritech-deep-learning-meetup-5/

Basic Knowledge of FCN models

- **The FCN model:** the last few fully connected layers are substituted with convolution layers which in turn enables a transform from a classification net into a spatial map. Moreover, if differentiable interpolation layers and spatial loss exists, end-to-end pixelwise learning will be possible.



Resource: E. Shelhamer, J. Long and T. Darrell. *Fully Convolutional Networks for Semantic Segmentation*. Retrieved May 15, 2018

Autoencoder in FCN

- a conventional convnet utilized for object detection and segmentation is expected to implement with autoencoder which consists of two parts, namely encoding part and decoding part
- The encoding part mainly transforms the image pixels into features and the decoding part functions vice versa
- In this project, the VGG-16 model is employed as the encoding part

Bilinear upsampling & Deconvolution

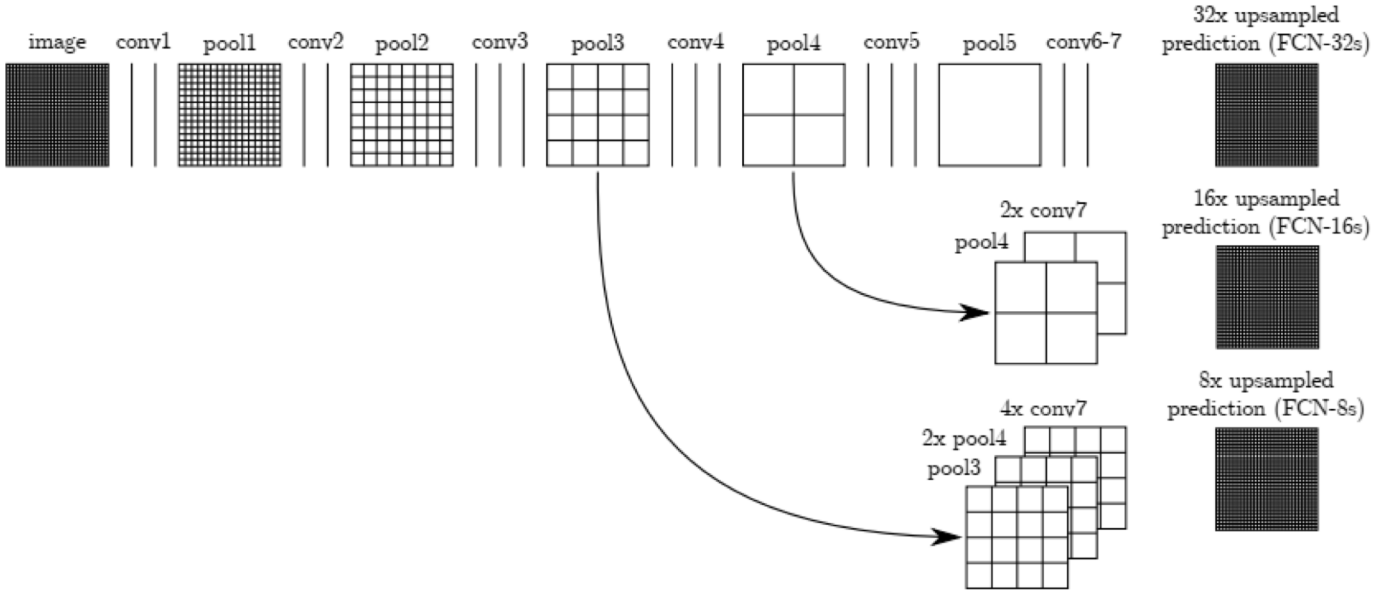
- the decoding part is implemented by bilinear upsampling, which is the reverse version of VGG-16 model which plays the role of deconvoluting the features generated by encoder to the image pixels again.
- In this project, due to the limitation of computing power, we freeze the encoding part parameters to be untrainable while activate the decoding part parameters to achieve a better performance based on limit resource we can access.

Combination and Refining

- Since the convnet for segmentation is defined to have feature hierarchy, it is natural to refine the spatial precision of the output at different layer
- 3 ways of manipulating upsampling step-size are proposed by Shelhamer et al. [7]
  - The first one named FCN-32s simply upsamples stride 32 predictions back to pixels in only one step
  - the second one FCN-16s makes combination of predictions from both the last layer and the 4^{th} pooling layer with upsampling stride 16
  - the third one FCN-8s upsamples at a shallower 3^{rd} pooling layer with stride 8, rather than predicting at 4^{th} pooling layers

Combination and Refining

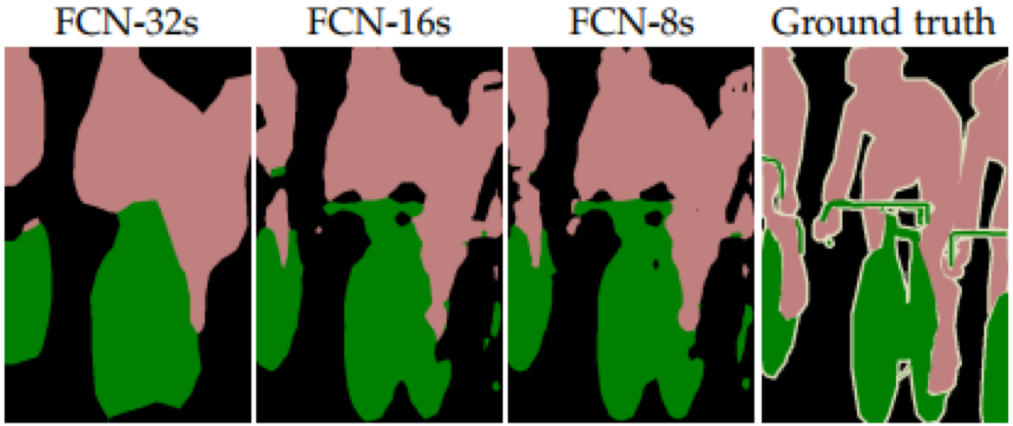
- Combination of predictions at different layer with diverse upsampling stride is shown in the figure below



Resource: E. Shelhamer, J. Long and T. Darrell. *Fully Convolutional Networks for Semantic Segmentation*. Retrieved May 15, 2018

Combination and Refining

- It is intuitive and also proved with experiments that the FCN-8s makes better predictions
- In our project, the FCN-8s method is implemented.



Resource: E. Shelhamer, J. Long and T. Darrell. *Fully Convolutional Networks for Semantic Segmentation*. Retrieved May 15, 2018

NYU

Project Procedure

Datasets, models and parameters in our project

Dataset used for Training and Validation

- Since long before the birth of FCN, the topic of object detection and segmentation has already aroused wide attention around the world.
- In order to achieve a better detection and segmentation result, many datasets intended for segmentation are generated and shared in public
- In this project, the famous PASCAL Visual Object Classes Challenge 2011 (PASCAL VOC2011) dataset is implemented for both training and validation

Dataset used for Training and Validation

- The PASCAL VOC2011 dataset contains 4 major classes of person, animal, means of transportation and furniture
- These major classes are further divided into 20 specific classes for accuracy
- In total, there are 11,530 images containing 27,450 ROI annotated objects and 5,034 segmentations [9] in PASCAL VOC 2011 dataset for both training and validation

Resource http://host.robots.ox.ac.uk/pascal/VOC/voc2011/index.html

Data Preprocessing

- In this project, before feeding images into the prototype model for training, several preprocessing such as Normalization is conducted to regularize and standardize the image pixels to prevent large coefficients dominate the small ones.

Model Summary

- The model applied in this project contains two parts, one is the encoder based on VGG-16 with untrainable parameters and the other is the deconvolution decoder by bilinear upsampling. The generated model contains 29 layers and several pooling layers.

Training/Validation Parameters

- `batch_size = 3 #which is recommended to be 10`
- `step_per_epoch = 1112`
- `step_validation = 1111`
- `epochs = 40 #which is recommended to be 100`

Model Summary

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 224, 224, 3)	0	
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792	input_1[0][0]
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928	block1_conv1[0][0]
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0	block1_conv2[0][0]
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856	block1_pool[0][0]
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147584	block2_conv1[0][0]
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0	block2_conv2[0][0]
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295168	block2_pool[0][0]
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590080	block3_conv1[0][0]
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590080	block3_conv2[0][0]
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0	block3_conv3[0][0]
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1180160	block3_pool[0][0]
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2359808	block4_conv1[0][0]
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2359808	block4_conv2[0][0]
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0	block4_conv3[0][0]
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2359808	block4_pool[0][0]
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2359808	block5_conv1[0][0]
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2359808	block5_conv2[0][0]
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0	block5_conv3[0][0]
block5_fc6 (Conv2D)	(None, 7, 7, 4096)	102764544	block5_pool[0][0]

Model Summary

block5_conv3 (Conv2D)	(None, 14, 14, 512)	2359808	block5_conv2[0][0]
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0	block5_conv3[0][0]
block5_fc6 (Conv2D)	(None, 7, 7, 4096)	102764544	block5_pool[0][0]
dropout_1 (Dropout)	(None, 7, 7, 4096)	0	block5_fc6[0][0]
block5_fc7 (Conv2D)	(None, 7, 7, 4096)	16781312	dropout_1[0][0]
dropout_2 (Dropout)	(None, 7, 7, 4096)	0	block5_fc7[0][0]
score_feat1 (Conv2D)	(None, 7, 7, 21)	86037	dropout_2[0][0]
score_feat2 (Conv2D)	(None, 14, 14, 21)	10773	block4_pool[0][0]
upscore_feat1 (BilinearUpSamplin	(None, 14, 14, 21)	0	score_feat1[0][0]
scale_feat2 (Lambda)	(None, 14, 14, 21)	0	score_feat2[0][0]
add_1 (Add)	(None, 14, 14, 21)	0	upscore_feat1[0][0] scale_feat2[0][0]
score_feat3 (Conv2D)	(None, 28, 28, 21)	5397	block3_pool[0][0]
upscore_feat2 (BilinearUpSamplin	(None, 28, 28, 21)	0	add_1[0][0]
scale_feat3 (Lambda)	(None, 28, 28, 21)	0	score_feat3[0][0]
add_2 (Add)	(None, 28, 28, 21)	0	upscore_feat2[0][0] scale_feat3[0][0]
upscore_feat3 (BilinearUpSamplin	(None, 224, 224, 21)	0	add_2[0][0]
activation_1 (Activation)	(None, 224, 224, 21)	0	upscore_feat3[0][0]

Total params: 134,362,751
Trainable params: 119,648,063
Non-trainable params: 14,714,688

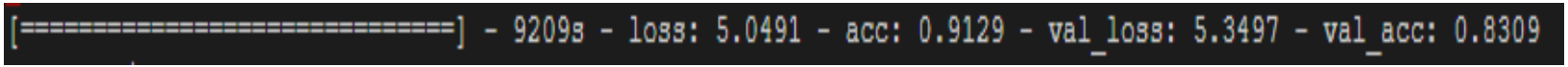
None

NYU

Results

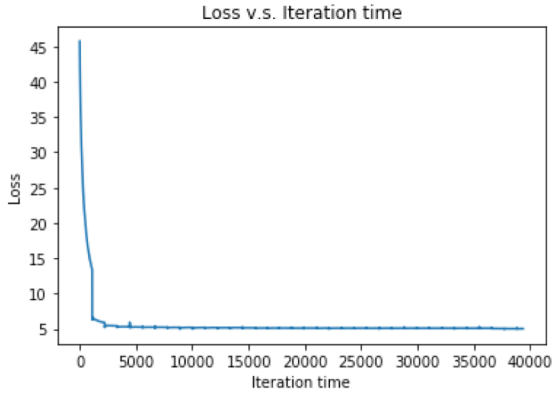
Training and Validation Accuracy

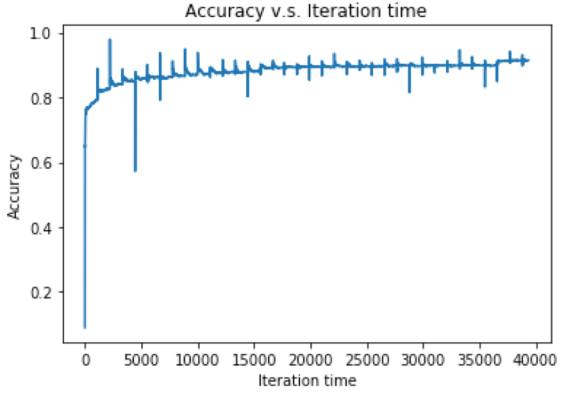
- After training at Google Cloud Platform (with 8 v-CPUs, 52GB memory) for 65 hours, the final accuracy of training and validation data can reach up to 91.29% and 83.09%, respectively.



Training summary

- Based on the log printed by nohup, we can review the loss and accuracy during the training process. From the figures, we can see the loss decreasing rapidly at first, then it keeps reducing gently, which is hard to track after several epochs (1112 iterations per epoch). Besides, the accuracy appears to be growing over time except for some singularity. These two curves meet our expectation. They partly imply that the model has captured some features from the data. That is, the parameters have been learnt.





Visualization the segmentation results

- In order to intuitively inspect the performance of detection and segmentation, several images are randomly picked to visualize the effect of segmentation. As is shown in figure below, the detection is generally authentic while the segmentation still needs more improvement.

NYU

Conclusion

Pros and Cons

- As can be vividly seen from the decreasing trend of training and validation loss, the generated model could achieve better performance when given more time
- With the current computing power and time limit, however, the number of epochs can hardly reach to what we have expected
- Fortunately, as is shown in figure above, the model is reasonable and effective in detecting and classifying the object while the segmentation part still needs to be improved

Acknowledgements

- Many people have made invaluable contributions both directly and indirectly to our project.
- We would like to express my warmest gratitude to Prof. Yao Wang, for her instructive suggestions when we are confronted with problems of computing power.
- At the same time, we are also grateful to Mr. Ish Kumar Jain for providing us with valuable advice and instructions concerning Google Cloud Platform.

NYU

THANK YOU FOR YOUR ATTENTION !

References

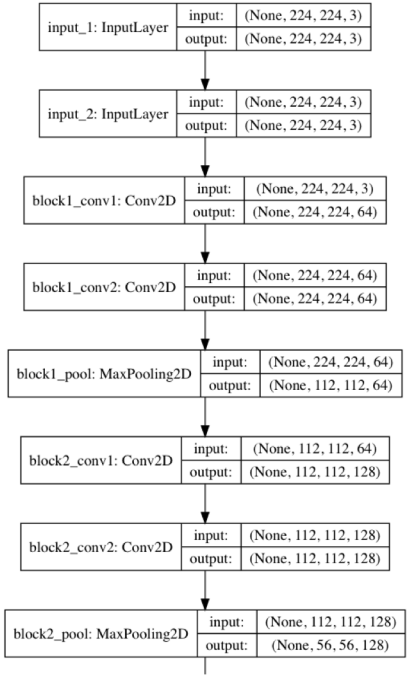
- [1] O. Matan, C. J. Burges, Y. LeCun, and J. S. Denker, “Multi-digit recognition using a space displacement neural network” in NIPS,1991, pp. 488–495. 2
- [2] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, “Backpropagation applied to handwritten zip code recognition” in Neural Computation, 1989.
- [3] R. Wolf and J. C. Platt, “Postal address block location using a convolutional locator network,” in NIPS, 1994, pp. 745–745.
- [4] F. Ning, D. Delhomme, Y. LeCun, F. Piano, L. Bottou, and P. E. Barbano, “Toward automatic phenotyping of developing embryos from videos,” Image Processing, IEEE Transactions on, vol. 14, no. 9, pp. 1360–1371, 2005.
- [5] D. C. Ciresan, A. Giusti, L. M. Gambardella, and J. Schmidhuber, “Deep neural networks segment neuronal membranes in electron microscopy images.” in NIPS, 2012, pp. 2852–2860.
- [6] Y. Ganin and V. Lempitsky, “N4-fields: Neural network nearest neighbor fields for image transforms,” in ACCV, 2014.

References

- [7] E. Shelhamer, J. Long and T. Darrell. Fully Convolutional Networks for Semantic Segmentation. Retrieved May 15, 2018, from https://people.eecs.berkeley.edu/~jonlong/long_shelhamer_fcn.pdf
- [8] Retrieved May 15, 2018 from https://blog.heuritech.com/2016/02/29/a-brief-report-of-the-heuritech-deep-learning-meetup-5/
- [9] Retrieved May 15, 2018 from http://host.robots.ox.ac.uk/pascal/VOC/voc2011/index.html
- [10] Krizhevsky, Alex, I. Sutskever, and G. E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems. 2012.
- [11] Jinghong Ju, keras-fcn, (2017), GitHub repository, Retrieved April 20, 2018 from https://github.com/JihongJu/keras-fcn
- [12] Jonathan Long*, Evan Shelhamer*, and Trevor Darrell. CVPR 2015 and PAMI 2016, GitHub repository, Retrieved April 20, 2018 from https://github.com/shelhamer/fcn.berkeleyvision.org

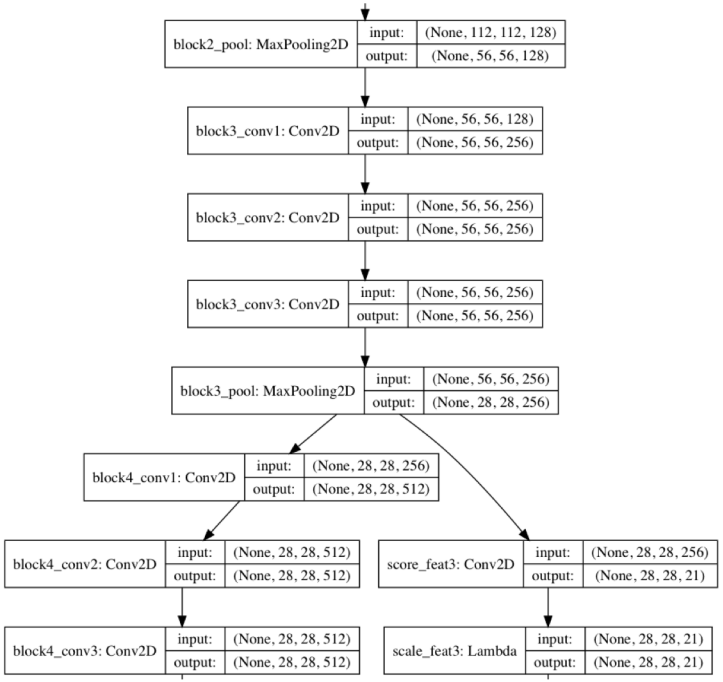
Appendix

- The plot of the model we used in this project



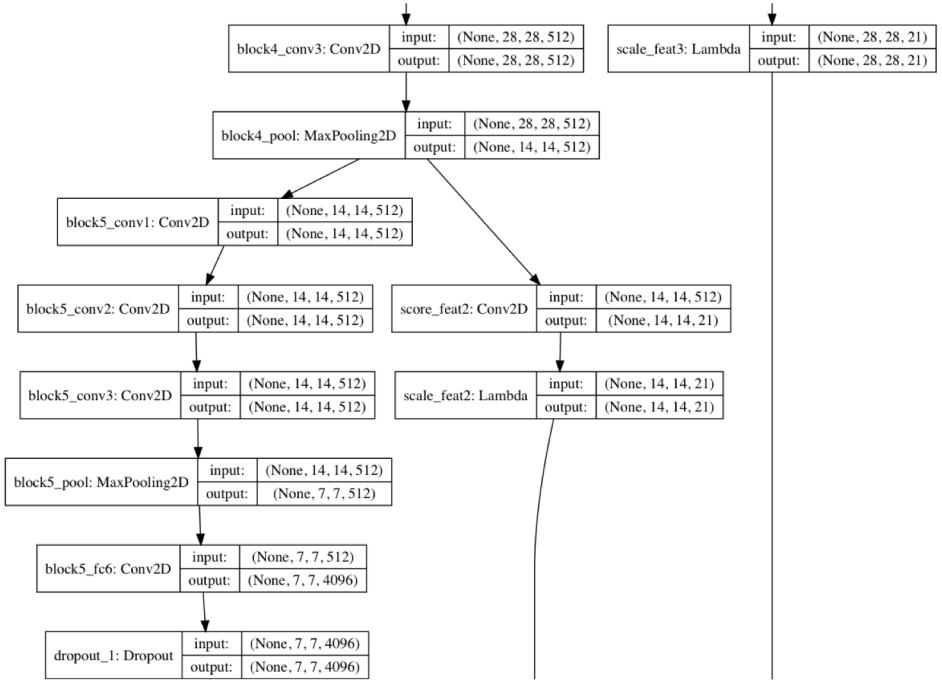
Appendix

- The plot of the model we used in this project



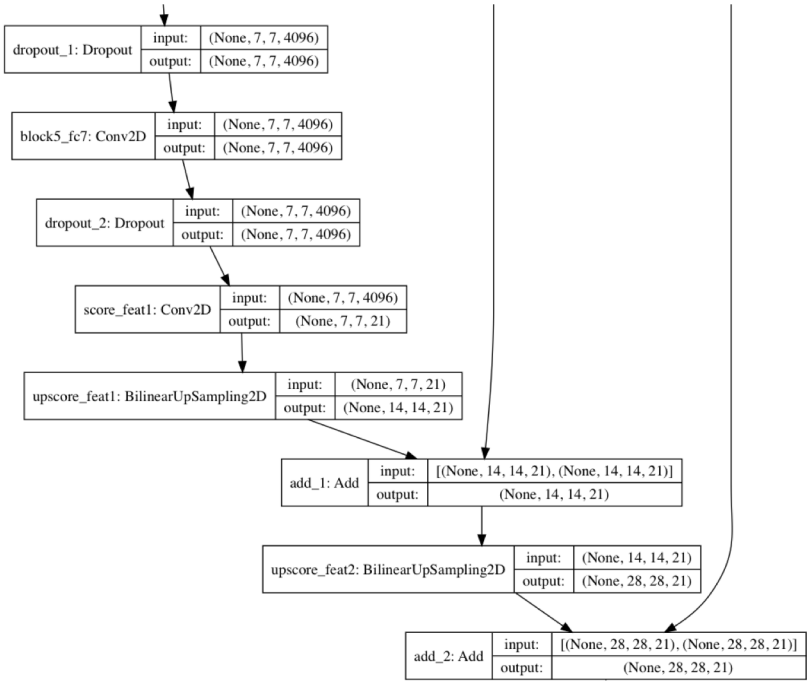
Appendix

- The plot of the model we used in this project



Appendix

- The plot of the model we used in this project



Appendix

- The plot of the model we used in this project