ASSIGNMENT 2 - COSC2673

SHEHZA HUSSAIN - S3811947

SHUBHANKAR SANJAY JAHAGIRDAR – S3793593

Introduction: In this project our goal is to develop a machine learning system that can classify histopathology images of colon cells. The dataset provided is “CRCHistoPhenotypes” dataset which consists of 27x27x3 RGB images of colon cells from 98 different patients. Using the data provided we have two main tasks: Firstly, we should classify images according to whether a given cell image represents a cancerous cell or not (isCancerous). Secondly, we need to classify images according to cell-type, like fibroblast, inflammatory, epithelial or others. Then we use the extracted information to train the isCancerous model to be trained and evaluated on the unlabelled data.

The provided dataset contains the information of the patient and the numerous instances of images along with labels to classify the information, these labels include 2 types of information cellType(4 types of cells) and isCancerous(0 if non-cancerous and 1 if cancerous). True labels for cancerous information has been provided for all 98 patients whereas for cellType only 39 patients.

TASK 01: Classify whether a given cell ‘is cancerous’ or ‘or not’ :

To perform this task, we have concatenated both the maindata.csv and the extradata.csv files as both the datasets have our target label column ‘iscancerous’. We start off by performing some EDA on this dataset, from which observe that our target label class has class imbalance with 0 - not cancerous instances being more than 1- is cancerous instances. We handled the class imbalance problem by setting the class_weights parameter to balanced. This enables us to use an evaluation metric called sparse categorical accuracy which we will be using to check if our data is being overfitted and to check the accuracy of our model. We split the data using random splitting technique into 3 sets, train, test, and validation data. However, we will be using macro averaged f1-score from sklearn as our base Evalution metric.

The very first model we have implemented is using VGG architecture - as it is one of the best models for image classification. First step in building any model is deciding any error metric and as we can see from our EDA the target class iscancerous is not balanced [Fig 1], therefore, we will be focussing on f1_score as our evaluation metric which we use from sklearn. However, we have handled the class imbalance problem by setting the class_weights parameter. This enables us to use an evaluation metric called sparse categorical accuracy which we will be using to check if our data is being overfitted and to check the accuracy of our model. We will be using sparse categorical cross entropy as our loss function. Some of the models that we have implemented include:

1] Grey scale model with VGG for binary classification: To begin with we start training our model on grey scale images because, using colour images increases the input number by 3 times corresponding to RGB values. This can cause 2 major issues: 1] curse of dimensionality: This says that large dimensional vectors are almost equally spaced and are difficult to separate by our classifier. 2] A larger number of parameters will be needed for a large network. Training larger number of parameters is difficult as it requires more data and iterations. This will lead to overfitting despite training well when compared to parameters of grey scale. However, our model [Fig 3] was overfitting with sparce categorical accuracy of 0.91 on testing and 0.83 on validation data (Hieu Minh Bui, 2016).

2] ResNet with RGB images for binary classification: We then moved onto implementing ResNet model on our data. The skip connections in ResNet eliminate the problem of vanishing gradient in deep neural networks by allowing this alternate shortcut path for the gradient to flow through.

Unfortunately, we see that Resnet does not give a good performance on our dataset. The results of our Resnet model [Fig 4] are: 0.97 for training and 0.62 for validation data. The model performs awfully bad with a large amount of overfitting (Mujtaba, 2020).

3] VGG with RGB images for binary classification: [Baseline model]: We finally implemented the VGG model with RGB images as our baseline model, 3 Conv2D layers with basic hyper parameters. On fitting the model, we see that it is overfitting, and this model might not generalise well to our unseen data. The performance of our baseline model has been recorded as [Fig 5] 0.9945 on training and 0.8673 on validation data.

On performing data augmentation and regularization we avoid overfitting present in our model. We use Relu as our activation function as this will output the input directly if it is positive, otherwise, it will output zero. Relu also eliminated the vanishing gradient problem. padding is set to same, and we have used he_normal as the kernel initializer. We are using a maxpooling layer then with a stride of 2*2 to avoid overfitting and achieving impressive performance. We are using dropout to set random inputs to 0, this helps in preventing overfitting. We choose 'Adam()' optimizer which works good with our non-linear activation function 'Relu' and sparse categorical accuracy as our evaluation metric with sparse categorical cross entropy as the loss function. The following results were obtained from or model after the hyper parameter tuning, sparce categorical accuracy of : 0.8749 on training and 0.8680 on validation data. [Fig 6] As we are focussing on f1-score as our evaluation metric we get the macro average f1-score from sklearns classification report as 0.86 (Popular Authors, 2018).

Performance Evaluation: As we have already stated, since we have found a class imbalance problem in this data, we will be using the macro average F1-scores to calculate the harmonic mean of the precision and recall to get the best performance measure of our model. We can observe from the [Table 1] the sparce categorical accuracy for our train and validation data. The table shows values of VGG model with RGB images, ResNet and VGG model with grey scale images. Since we are passing all the features of our data set to all these models, we can state that in comparison to other models, *VGG model for RGB images* performs better with a macro averaged f1-score of 86%, we can use this as our final model to do the predictions on the test data.

TASK 02: Classify images according to cell-type:

In order to perform this task we are using only the maindata.csv, our target label class is the cell type. There are 4 cell types fibroblast, inflammatory, epithelial and others. From the EDA of this dataset we learn that the target class (cell type) is not balanced [Fig 2], hence we will be using f1-score from sklearn as our evaluation metric for the evaluating the performance of our model. We have handled the class imbalance problem by setting the class_weights parameter to balance the class imbalance. This enables us to use another evaluation metric called sparse categorical accuracy which we will be using to check if our data is being overfitted and to check the accuracy of our model. We will be using sparce categorical cross entropy as our loss function.

We then split the data using random splitting technique into train, test and validation data.

We have implemented a VGG 16 model - as it is one of the best models for image classification. As mentioned above the first step in building any model is deciding any error metric and as we can see from our EDA [Fig 2] the target class cell type is not balanced, therefore, we will go with macro averaged f1_score from sklearn. We will use Sparse categorical accuracy as our metric for determining if there is any overfitting and sparse categorical cross entropy as our loss function (Hieu Minh Bui, 2016).

Baseline model: We implement VGG model with basic hyper parameters and observe that the model (K. Sirinukunwattana, May, 2016)overfits to a great extent. We start off with 3 Conv2D layers. We choose 'Adam()' optimizer with a learning rate of 0.001 which works good with our non-linear activation function 'Relu'. We obatin 0.8421 on training and 0.6080 on validation data.

Overfitting can be avoided by using techniques such as data augmentation, regularization and so onn. We use 2 Conv2D layers and apply some data augmentation which will increase the data we have, thus eliminating overfitting. We use a lamda value of ----- with a learning rate of ----

We use Relu as our activation function as this will output the input directly if it is positive, otherwise, it will output zero. Relu also eliminated the vanishing gradient problem. padding is set to same, and we have used he_normal as the kernel initializer. We are using a maxpooling layer then with a stride of 2*2 to avoid overfitting and achieving impressive performance. We are using dropout to set random inputs to 0, this helps in preventing overfitting. We choose 'Adam()' optimizer which works good with our non-linear activation function 'Relu'. The macro average f1-score obtained was 62% with sparce categorical accuracy of 0.7409 for training and 0.7091 on validation data. [Fig 6]

Performance Evaluation: [Cell type prediction] Due to the class imbalance problem in this dataset, we will be using the macro average F1-scores to calculate the harmonic mean of the precision and recall. From [Table 2] we can observe the values for VGG model with default hyper parameters and VGG model with hyper parameter tuning. As we are passing the same data to both the models, we can say from the values of train score and validation score that the VGG model with hyper parameter tuning performs best on our data and can be used to predict on unseen data. This will be considered as our best and final model with a macro average f1-score of 62%.

INDEPENDENT EVALUATION:

Comparing the architecture's: Recent studies have shown that deep learning methods produce promising results on many histopatho-logical image datasets. To begin with let's talk about the architecture of the research paper [Table 3] being reviewed along with the architecture of our VGG-16 model. They have used neighbouring ensemble predictor (NEP) along with a standard SoftMax CNN. The proposed approach for detection and classification uses a sliding window to train the networks on small patches instead of the whole image. Refer Table 3 for in depth explanation about the layers.

On the other hand we have used VGG – 16 models for performing both our tasks [Fig 7, Table II] (predicting iscancerous and cell type) It makes the improvement over AlexNet by replacing large kernel-sized filters with multiple 3×3 kernel-sized filters one after another. Our network consists of Conv2D layers, maxpooling layer and fully-connected layers. We use 'Adam()' as our optimizer with macro avg f1-score as our evalution metric. For in dept understand refer Table 04.01 and 04.02

Comparing performance: In the paper being reviewed they have calculated the F1 score for each class of nucleus and their average weighted by the number of nucleus samples to summarize the overall classification performance. We also considered an area under the receiver operating characteristic curve for multiclass classification (multiclass AUC)[Table 3][Table 5, 6, 7].

Talking about our model (VGG), as mentioned already we have used Maco avg f-1 score to understand how well our model performs along with the learning curves to observe any obvious trend and overfitting if any. We use f1-score as we observe data imbalance of our target variable. The results are given in the form of a table for both is cancerous and cell type prediction [Table 1][Table 2].

Figures:

Figure: 01 Is cancerous data distribution.

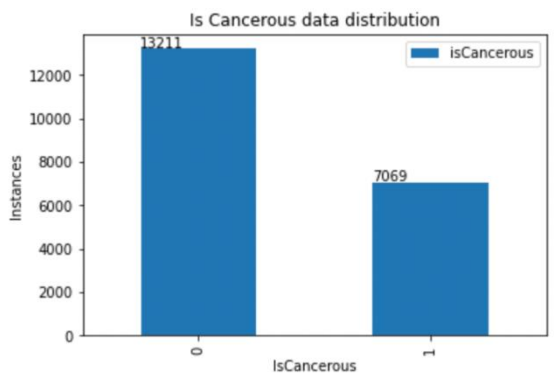
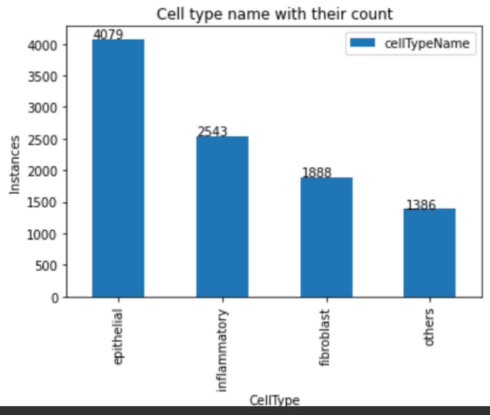


Figure: 02: Cell type value count



3.

Figure: 03 VGG – Grey scale

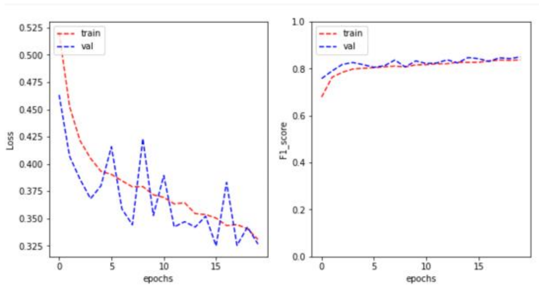


Figure: 04 Figure: 02 ResNet – RGB

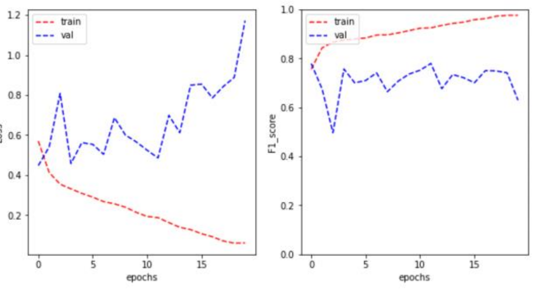


Figure: 05 VGG – RGB [Baseline model].

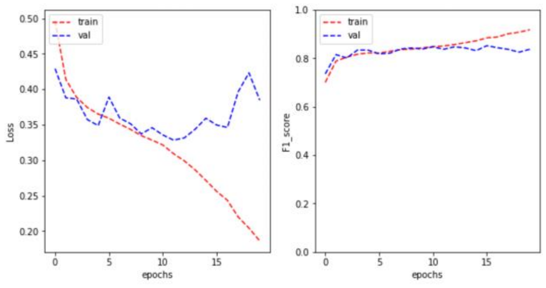
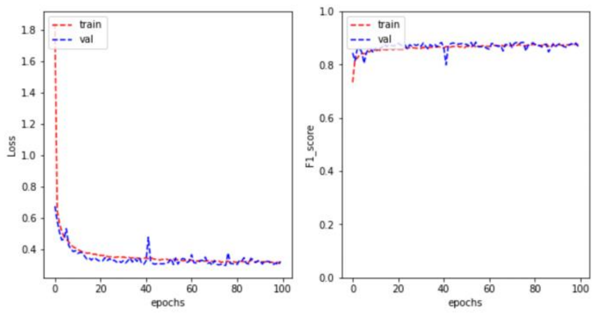


Figure: 06 VGG – RGB



TABLES:

Table : 01 <u>**Is cancerous models evaluations**</u>

Model	Training score	Testing score	Macro avg f1
VGG - grey scale	91%	83%	84%

ResNet - RGB	97%	62%	-
VGG - RGB	87%	86%	86%

Table : 02 **<u>Cell type prediction model evaluation</u>**

Model	Training score	Testing score	Macro avg f1
VGG – Base model	84%	60%	-
VGG – Final model	74%	71%	62%

Table : 03 **<u>Architecture for SC-CNN and SoftMax CNN</u>**

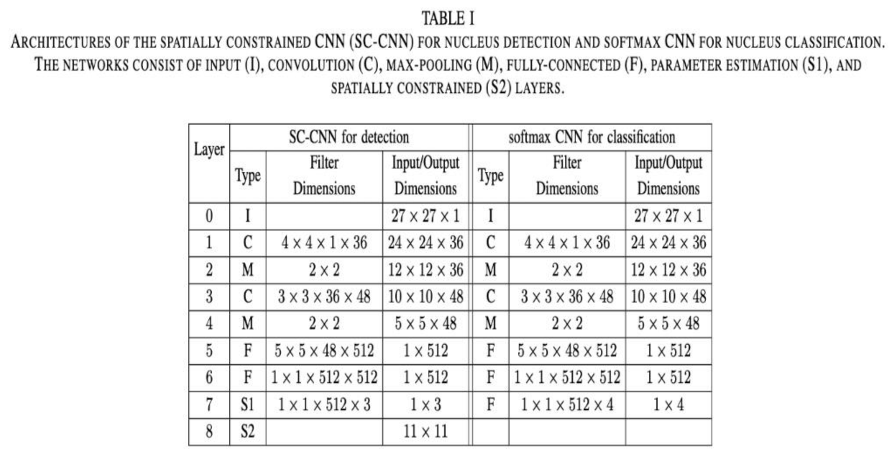
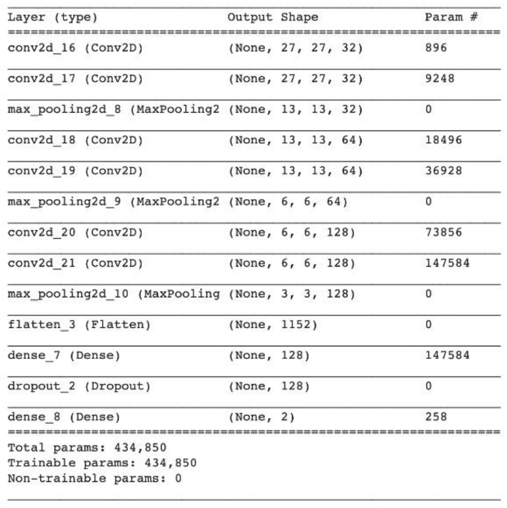


TABLE I
ARCHITECTURES OF THE SPATIALLY CONSTRAINED CNN (SC-CNN) FOR NUCLEUS DETECTION AND SOFTMAX CNN FOR NUCLEUS CLASSIFICATION. THE NETWORKS CONSIST OF INPUT (I), CONVOLUTION (C), MAX-POOLING (M), FULLY-CONNECTED (F), PARAMETER ESTIMATION (S1), AND SPATIALLY CONSTRAINED (S2) LAYERS.

Layer	SC-CNN for detection			softmax CNN for classification		
	Type	Filter Dimensions	Input/Output Dimensions	Type	Filter Dimensions	Input/Output Dimensions
0	I		27 × 27 × 1	I		27 × 27 × 1
1	C	4 × 4 × 1 × 36	24 × 24 × 36	C	4 × 4 × 1 × 36	24 × 24 × 36
2	M	2 × 2	12 × 12 × 36	M	2 × 2	12 × 12 × 36
3	C	3 × 3 × 36 × 48	10 × 10 × 48	C	3 × 3 × 36 × 48	10 × 10 × 48
4	M	2 × 2	5 × 5 × 48	M	2 × 2	5 × 5 × 48
5	F	5 × 5 × 48 × 512	1 × 512	F	5 × 5 × 48 × 512	1 × 512
6	F	1 × 1 × 512 × 512	1 × 512	F	1 × 1 × 512 × 512	1 × 512
7	S1	1 × 1 × 512 × 3	1 × 3	F	1 × 1 × 512 × 4	1 × 4
8	S2		11 × 11			

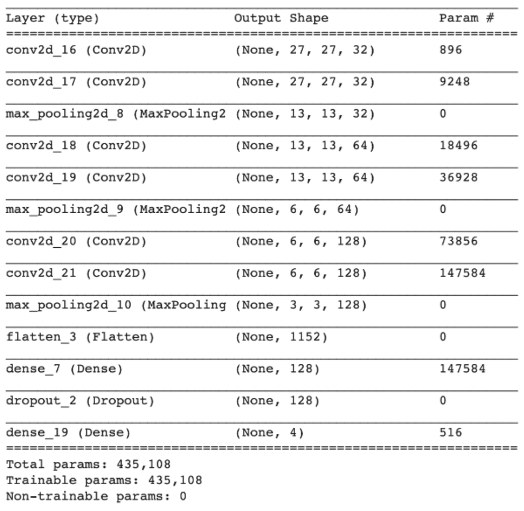
<u>Architecture of VGG for Iscancerous prediction and for cell type prediction:</u>

Table : 04.01



```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_16 (Conv2D)           (None, 27, 27, 32)        896
_________________________________________________________________
conv2d_17 (Conv2D)           (None, 27, 27, 32)        9248
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 13, 13, 32)        0
_________________________________________________________________
conv2d_18 (Conv2D)           (None, 13, 13, 64)        18496
_________________________________________________________________
conv2d_19 (Conv2D)           (None, 13, 13, 64)        36928
_________________________________________________________________
max_pooling2d_9 (MaxPooling2 (None, 6, 6, 64)          0
_________________________________________________________________
conv2d_20 (Conv2D)           (None, 6, 6, 128)         73856
_________________________________________________________________
conv2d_21 (Conv2D)           (None, 6, 6, 128)         147584
_________________________________________________________________
max_pooling2d_10 (MaxPooling (None, 3, 3, 128)         0
_________________________________________________________________
flatten_3 (Flatten)          (None, 1152)              0
_________________________________________________________________
dense_7 (Dense)              (None, 128)               147584
_________________________________________________________________
dropout_2 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_8 (Dense)              (None, 2)                 258
=================================================================
Total params: 434,850
Trainable params: 434,850
Non-trainable params: 0
_________________________________________________________________
```

Table : 04.02



```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_16 (Conv2D)           (None, 27, 27, 32)        896
_________________________________________________________________
conv2d_17 (Conv2D)           (None, 27, 27, 32)        9248
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 13, 13, 32)        0
_________________________________________________________________
conv2d_18 (Conv2D)           (None, 13, 13, 64)        18496
_________________________________________________________________
conv2d_19 (Conv2D)           (None, 13, 13, 64)        36928
_________________________________________________________________
max_pooling2d_9 (MaxPooling2 (None, 6, 6, 64)          0
_________________________________________________________________
conv2d_20 (Conv2D)           (None, 6, 6, 128)         73856
_________________________________________________________________
conv2d_21 (Conv2D)           (None, 6, 6, 128)         147584
_________________________________________________________________
max_pooling2d_10 (MaxPooling (None, 3, 3, 128)         0
_________________________________________________________________
flatten_3 (Flatten)          (None, 1152)              0
_________________________________________________________________
dense_7 (Dense)              (None, 128)               147584
_________________________________________________________________
dropout_2 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_19 (Dense)             (None, 4)                 516
=================================================================
Total params: 435,108
Trainable params: 435,108
Non-trainable params: 0
```

Table : 05

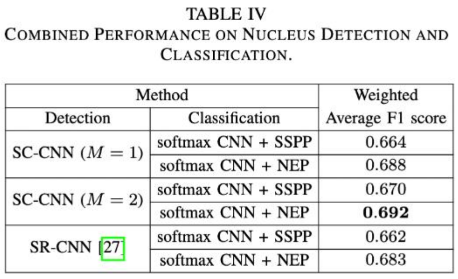
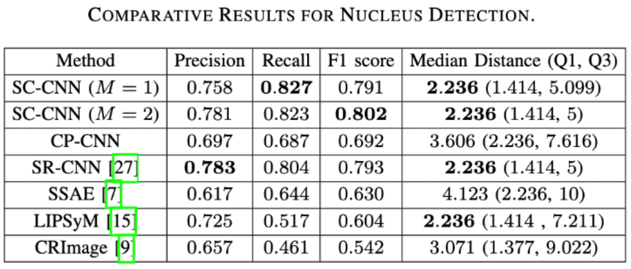


TABLE IV
COMBINED PERFORMANCE ON NUCLEUS DETECTION AND CLASSIFICATION.

Method		Weighted
Detection	Classification	Average F1 score
SC-CNN ($M = 1$)	softmax CNN + SSPP	0.664
	softmax CNN + NEP	0.688
SC-CNN ($M = 2$)	softmax CNN + SSPP	0.670
	softmax CNN + NEP	**0.692**
SR-CNN [27]	softmax CNN + SSPP	0.662
	softmax CNN + NEP	0.683

Table : 06



COMPARATIVE RESULTS FOR NUCLEUS DETECTION.

Method	Precision	Recall	F1 score	Median Distance (Q1, Q3)
SC-CNN ($M = 1$)	0.758	**0.827**	0.791	**2.236** (1.414, 5.099)
SC-CNN ($M = 2$)	0.781	0.823	**0.802**	**2.236** (1.414, 5)
CP-CNN	0.697	0.687	0.692	3.606 (2.236, 7.616)
SR-CNN [27]	**0.783**	0.804	0.793	**2.236** (1.414, 5)
SSAE [7]	0.617	0.644	0.630	4.123 (2.236, 10)
LIPSyM [15]	0.725	0.517	0.604	**2.236** (1.414 , 7.211)
CRImage [9]	0.657	0.461	0.542	3.071 (1.377, 9.022)

Table : 07

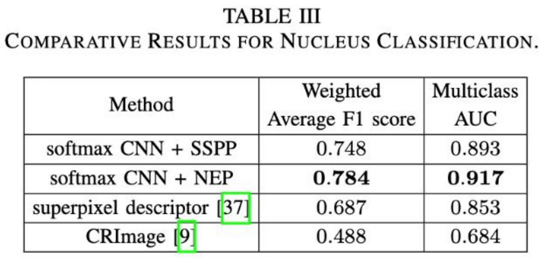


TABLE III
COMPARATIVE RESULTS FOR NUCLEUS CLASSIFICATION.

Method	Weighted Average F1 score	Multiclass AUC
softmax CNN + SSPP	0.748	0.893
softmax CNN + NEP	**0.784**	**0.917**
superpixel descriptor [37]	0.687	0.853
CRImage [9]	0.488	0.684

References

Hieu Minh Bui, M. L. (2016 , July 27). *ieeexplore.* Retrieved from https://ieeexplore.ieee.org/document/7562656: https://ieeexplore.ieee.org/document/7562656

K. Sirinukunwattana, S. E. (May, 2016). Locality Sensitive Deep Learning for Detection. *IEEE Transactions on Medical Imaging*.

Mujtaba, H. (2020, September 28). *mygreatlearning.* Retrieved from https://www.mygreatlearning.com/blog/resnet/: https://www.mygreatlearning.com/blog/resnet/

Popular Authors. (2018, November 20). *https://neurohive.io/en/popular-networks/vgg16/.* Retrieved from https://neurohive.io/en/popular-networks/vgg16/.