Face Mask Detection by Convolutional Neural Networks of Different Architectures

Boyu Shen
bshen27@wisc.edu

Zining Tan
ztan46@wisc.edu

Ruiting Tong
rtong5@wisc.edu

Abstract

Detecting facemasks could be useful during this time since COVID-19 virus is spreading rampantly across the world. Facemask detecting programs enable the governors to better monitor whether people are wearing facemasks in the public area. Our motivation is to build a model to detect whether people are wearing facemask in the photo so that we can help preventing the spread of COVID-19. Governors can connect this model to the monitors on roads or in public areas such as shopping mall and give warning to those who do not wear masks. The dataset we used in this project is from Kaggle. It consists of 7553 images divided into two folders by whether the person in the photo wear masks. We constructed five models using five different architectures of convolutional neural network, including ResNet34, AlexNet, LeNet5, VGG16, and all-convolutional network. The results show that AlexNet and VGG16 are outperforming other models. The VGG16 model have the highest accuracy 94.17% though the runtime is very long. The AlexNet model's accuracy is slightly lower than VGG16 but it runs much faster than VGG16.

1. Introduction

Nowadays. COVID-19 spread rampantly across the world. The government and communities have imposed vary restriction on people's daily life, the most common of which is the mandatory requirement to wear face masks in public areas. However, there are always failure of compliance, we can still see people not wearing masks on the road and in the buildings. This is very dangerous for others since many of the COVID-19 carriers may not have symptoms appears, but they can still spread the virus by coughing, sneezing, or even speaking at close range. Therefore, the communities need to figure out a way to monitor the wearing of facemasks.

In general, the government and community can hire people checking for mask at the entrance of buildings. However, this cannot guarantee all the places. For example, our campus have mask check in the entrance. However, it cannot monitor whether the students and employees still wear masks after they get into the building and go into the offices or classrooms. Furthermore, it is impractical for companies and communities to hire more people to monitor the face mask wearing in public area, since it is a large cost for them. We believe that we can design a model to detect whether people are wearing masks, we can connect it to the cameras in the shopping malls, or on the road to help the governors detect whether people are wearing masks. Warning to those who are not wearing masks can be given by public speakers. This can help the government and communities save the labor cost and most importantly, it can help stopping the spread of COVID-19.

Figure 1. Example Pictures with Mask,Source:https://www.kaggle.com/alincijov/face-mask-detection-pytorch-cnn

Figure 2. Example Pictures with Mask,Source:https://www.kaggle.com/alincijov/face-mask-detection-pytorch-cnn

For this project, our team aims to detect whether people in photo are wearing facemasks based on the data we obtain from Kaggle using the deep learning method. Fig-

ure.1 and Figure.2 are two examples of the pictures with masks in our dataset.Figure.3 and Figure. 4 are examples of pictures without masks. We will divide the data into two classes, with mask and without mask. After we obtain the training accuracy of the models we generate, we will focus on the model with highest accuracy. Convolutional neural network (CNN) will be the method we are going to use for our model. We will compare the performance of different CNN architectures including AlexNet, LeNet5, VGG16, all-convolutional network, and ResNet34.

Figure 3. Example Pictures without Mask,Source:https://www.kaggle.com/alincijov/face-mask-detection-pytorch-cnn

Figure 4. Example Pictures without Mask,Source:https://www.kaggle.com/alincijov/face-mask-detection-pytorch-cnn

2. Related Work

Various convolutional neural networks are employed in a number of studies to finish the face mask detection task. One related study whose objective is to annotate and localize the medical face mask(identify the location of the bounding box of mask), utilized the ResNet-50 deep transfer learning model to do the feature extraction and YOLO v2 deep network to detect the medical mask [5]. According to Loey et al., the dataset were a combined dataset of two smaller data from different sources, which would add to the randomness.

For the training part, data augmentation is used at first to artificially increase the diversity of the datasets for training detectors, which would improve the performance of the detector in training. Then in their proposed model, a residual neural network (ResNet-50) is used as a deep transfer model for feature extraction. There are 16 residual bottleneck blocks, and each block has convolution size 1x1,3x3,and1x1 with feature maps (64, 128, 256, 512, 1024). They also utilize YOLO v2 which contain a few convolutional layers, transform layer, and finally output layer to detect the face mask. The transform layer extracts activations of convolutional layer and improves the steadiness of the deep neural network. The transform layer converts the bounding box forecast to be in outlines of the target box. The locations of pure bounding box is produced by the output layer.

The loss function calculates the mean squared error loss between predicted bounding box and the target, and it consists of three parts: Localization Loss, Confidence Loss and Classification Loss. Localization loss measures error between the target and the predicted bounding box.Confidence loss computes the confidence score of error. And the classification loss measures the error between the class conditional probabilities for each class in grid cell i.They have reached a average precision of 81% using Adam optimizer.

Other related work include the study by Guan Wang and Jun Gong [8]. They proposed an improved cross-connected multilayer LeNet-5 Convolutional Neural Network model do the facial expression recognition work. They claimed their model is more robust and adaptable to real-life environment where the lack of expression information in the occlusion area may influence the model performance. This study adds a convolution layer and a pooling layer based on the LeNet-5 model. The implicit features are extracted by using the trainable convolution kernel, and the extracted implicit features are reduced by the pooling layer. The Softmax classifier is used for classification and recognition. They also verified the occlusion robustness of their improved method in the study.

We also used Lenet-5 and Resnets Models in our project. Compared with the related work, our model architectures are easier to understand and easier to implement. Since our target is a binary variable and the examples in our dataset are less diverse with most pictures showing the whole face, our task at hand is easier to finish and we do not have to use complex loss functions.

3. Proposed Method

3.1. Pre-processing

The images we collect naturally provide us with 3 color channels: red, green, and blue. Each channel can be represented by a matrix with dimensions equal to the number of pixels in the length and width of its corresponding image. While we may directly use all three color channels as our features, we can better accommodate some CNN architectures by first transforming the images to black and white. We also reshape the images into equal sizes and standardize

their corresponding matrices around mean 0 so that each pixel has the same distribution as others [**?**]. The transformed data are then randomly split into training, validation, and test sets. We are not applying further perturbation to the training set.

3.2. Feature Normalization

In our project, Batch normalization (Batchnorm), a common technique for feature normalization, is applied to almost every hidden layer before activation in each architecture. The only exception is ResNet34, where Batchnorm is incorporated into the network structure itself. Batchnorm is designed to stabilize weight updates in deep neural networks and improve performance [1, p.310]. Suppose that the inputs of a layer are given by a matrix $\boldsymbol{H}$, its mean and standard deviation being $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$; then the idea of Batchnorm is to first transform $\boldsymbol{H}$ by

$$\boldsymbol{H'} = \frac{\boldsymbol{H} - \boldsymbol{\mu}}{\sqrt{\boldsymbol{\sigma^2} + \boldsymbol{\epsilon}}}, \tag{1}$$

where ϵ is a small positive term to prevent division by 0 [1, p.310]. Then $\boldsymbol{H'}$ is replaced by $\boldsymbol{\gamma H'} + \boldsymbol{\beta}$, where γ and β are trainable parameters, to "maintain the expressive power of the network" [1, p.312].

The purpose of adding a Batchnorm layer to each hidden layer is to boost the performance of each CNN architecture we experiment on, and ensure that no architecture is taking advantage of it. Then pairwise comparisons may become more meaningful.

3.3. Optimization Strategy

For each architecture, we optimize the loss function by stochastic gradient descent (SGD) with a fixed momentum and an adaptive learning rate. Let V_{t-1} be the velocity at time $t-1$, g be the gradient, α be the learning rate, and β be the parameter of momentum. Then the velocity V_t at time t is defined by $V_t = \beta V_{t-1} - \alpha g$ [2]. This reduces the noise of gradients and makes convergence faster [2]. The learning rate is reduced whenever the loss function peaks.

3.4. Architectures

3.4.1 LeNet5

LeNet-5 is a simple convolutional neural network proposed by Yann Lecun et al. in 1989 [6]. It includes two convolution layers, two pooling layers and a fully connected layer. In the first convolution layer, It takes the input images of 32 pixels by 32 pixels and convert it into multiple channels of feature maps. In the following pooling layer, it subsamples from the pixels in multiple channels to create smaller sized feature maps. After two convolution layers and pooling layers, multilayer perceptron would be utilized to create a fully connected layer. Basically the the loss function would be mean squared error, but nowadays softmax and cross entropy are more commonly used. See figure.5 for a visualization of the LeNet5 Model.

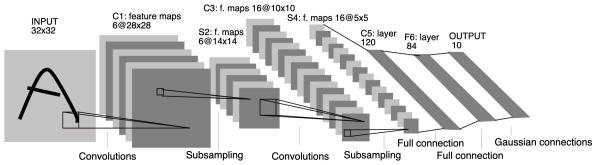

Figure 5. LeNet-5 Artichitecture,Source:`https://paperswithcode.com/method/lenet`

3.4.2 All-convolutional Network

Springenberg, Jost Tobias, et al suggest in their paper that pooling is not a necessary component of a convolutional neural network. They found that their CNN without subsampling layers "matches or slight outperforms the state of art on CIFAR10 and CIFAR100" [3]. In light of this, we construct an all-convolutional network that consecutively double the number of channels and halve the size of images. Figure. 6 displays an example of the All-convolutional Network.

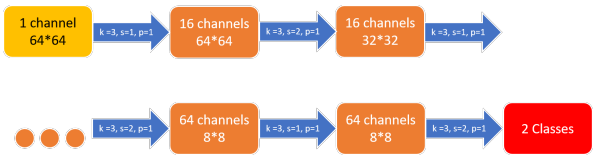

Figure 6. Structure of our all-convolutional network, made by Ruiting Tong

3.4.3 AlexNet

We also adopted AlexNet to train our model. AlexNet is one of the convolutional neural network (CNN) architecture which consists of five convolutional layers and three fully-connected layers. AlexNet allows multi-GPU training, one GPU runs the layer-parts at the top of the figure while the other runs another half layers at the bottom. This can not only train larger models but also shorten the training time. Figure.7 shows an overview of Alexnet model.

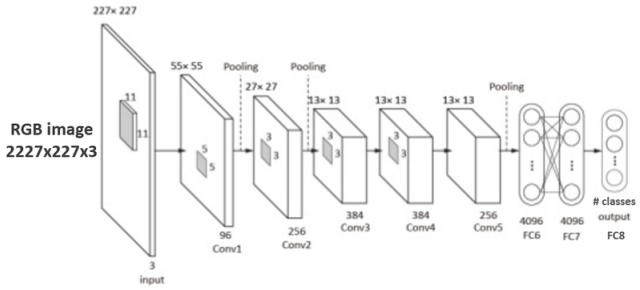

Figure 7. Structure of AlexNet. Source: https://www.researchgate.net/figure/AlexNet-CNN-architecture-layers_fig1_318168077

3.4.4 VGG16

VGG16 was invented by Simonyan, Karen, and Andrew Zisserman, who tried to improve CNN performance by increasing its depth while decreasing the kernel sizes at each convolutional layer[9]. It turned out to be a success, and its architecture is shown below. Figure.8 is a visualization of the VGG-16 model.

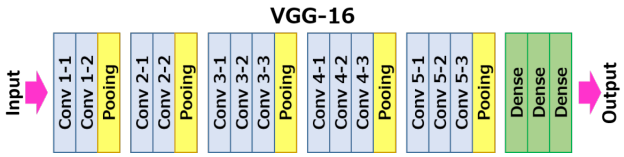

Figure 8. Structure of VGG16. Source: https://neurohive.io/en/popular-networks/vgg16/

3.4.5 ResNet34

ResNets are residual learning framwork which allow skip connections. Figure.9 shows the process of a layer, where x is the layer input .When all weights and bias are zero, the output becomes identitiy function which gives the possibility to learn identity functions and skip layers that are not useful. Thus, in Resnets we can implement very deep architectures.

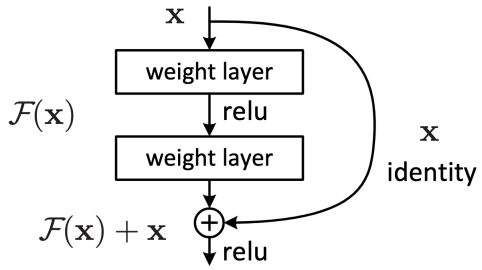

Figure 9. Resnet-34 Architecture Source:https://neurohive.io/en/popular-networks/resnet/

4. Experiments

4.1. Dataset

The face mask images are available on https://www.kaggle.com/omkargurav/face-mask-dataset [4]. There are a total of 7553 images of different dimensions, 3725 labeled as "masked" and 3828 "unmasked". Stratifying on the label proportions, we randomly select 60% of them as training samples, 30% as test samples, and 10% as validation samples. The models are trained with the training samples only. We set 50 epochs for training each architecture and 256 batches for each epoch. Accuracies on the training set and the validation set are reported at the end of each epoch. Eventually, each model is evaluated on the test set, earning a final accuracy score.

Since the images do not share a same shape, we reshape all of them to 64 pixels by 64 pixels for all architectures except LeNet5, whose structure begins with a 32 by 32 input. Images are converted to black and white if the corresponding architecture starts with one input channel. We assume that the mean value and the standard deviation of each channel are equal to 0.5, and standardize the images using this value. During the back propagation, the initial learning rate of SGD is set to 0.1, and the parameter of momentum is 0.9. Whenever there is a peak of loss, the scheduler will decrease the learning rate by a factor of 10. The data preparation for the five different architectures can be summarized by the following table.

Architecture	Image Type	Image Size
LeNet5	BW	32×32
All-convolutional network	BW	64×64
AlexNet	RGB	64×64
VGG16	RGB	64×64
ResNet34	RGB	64×64

Table 1. the summary of experiment setup

4.2. Software

The PyTorch library has provided us with a variety of tools and functions including image transformers, data load-

ers, and model builders [10]. We also owe to Matplotlib for the visualization of the training process [11].

5. Results and Discussion

We shall first present the classification accuracies of the proposed architectures:

Architecture	Test accuracy
LeNet5	86.00%
All-convolutional network	92.67%
AlexNet	94.04%
VGG16	94.17%
ResNet34	92.54%

Table 2. Classification accuracies for different architectures.

LeNet5 is clearly outperformed by other CNN architectures in our binary classification task, and as can be seen from Figure 10 below, this model has been trained sufficiently at epoch 50, with a training accuracy close to 100%. Since LeNet5 is a simple CNN structure [7] designed to work on images of only 32 pixels by 32 pixels with just one input channel, a lot of useful information has to be cropped from the images. We have also observed some pictures where the mask only takes up a very small area, and chances are that it becomes completely blurred and unidentifiable after image reshaping, thus placing LeNet5 in a disadvantage compared to other architectures. However, the advantage of being a simple model is that it requires a short time to train. In our case, it takes only 21.36 minutes for LeNet5 to finish all epochs.

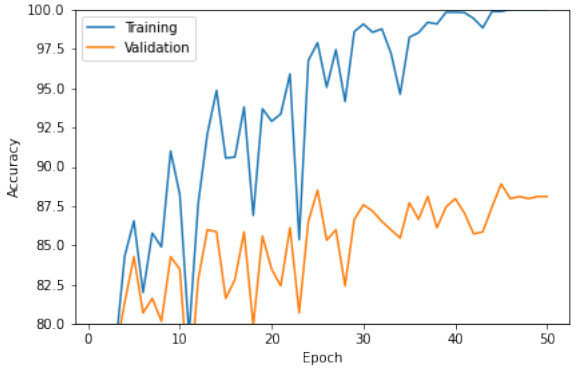

Figure 10. LeNet5 Training and Validation Accuracy vs. Epoch.

Our all-convolutional network performs reasonably well given that it receives only one channel of color, achieving an accuracy of 92.67%. An interesting observation is that learning takes place very quickly for this specific model. In Figure 11, the training accuracy quickly spikes to 100% in 25 epochs, and the performance levels off. Observe that overfitting is obvious from both Figure 10 and 11, and we suggest that using dropout method may help alleviate overfitting and further boost classification accuracy.

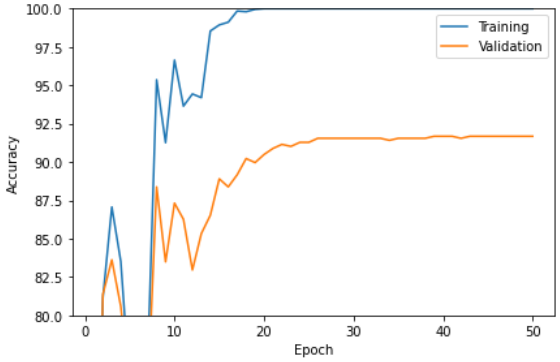

Figure 11. All-convolutional Network Training and Validation Accuracy vs. Epoch.

AlexNet displays an outstanding performance on the classification task. Despite being 0.13% in test accuracy compared to VGG, it is almost ten times faster. Being implemented and trained on the same laptop, it took AlexNet only 53.78 minutes to finish 50 epochs, while it took VGG16 515.29 minutes. Furthermore, from the plot of its training and validation accuracy (Figure 12), we observe less overfitting than the previous two architectures. We speculate that the two dropout layers in the fully connected portion of AlexNet are contributing to this result.

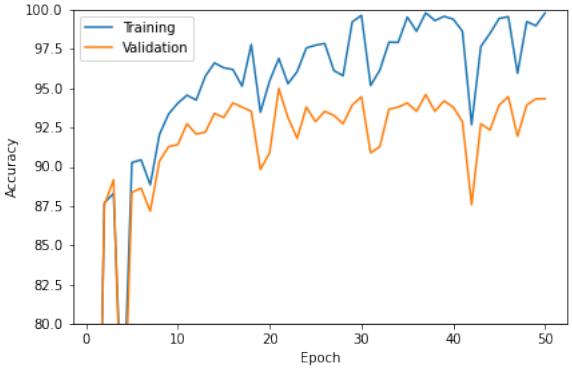

Figure 12. AlexNet Training and Validation Accuracy vs. Epoch.

VGG16 attains the highest test accuracy of 94.17%, but its large set of parameters result in a long training time. Yet a high performance is not the only advantage of VGG16 for this specific task. Overfitting is minimally seen in the training process, which is reflected by the close distance between the curve of validation accuracy and that of training accuracy in Figure 13. Judging from the trend of curves, we believe it is possible that given more epochs, the model will be trained even better. Another surprising observation is a sudden plummet of accuracy and rise of loss at the 11th

epoch. Although a similar pattern can be seen other architectures during early epochs, the drop of accuracy to 50% in VGG16 training basically means everything is unlearnt to a state of random guess, which is beyond the scale of large fluctuations in previous models. Our opinion is that the sudden increase in loss is caused by a large learning rate in the beginning. The updates of weights may have significantly overshot their intended values.

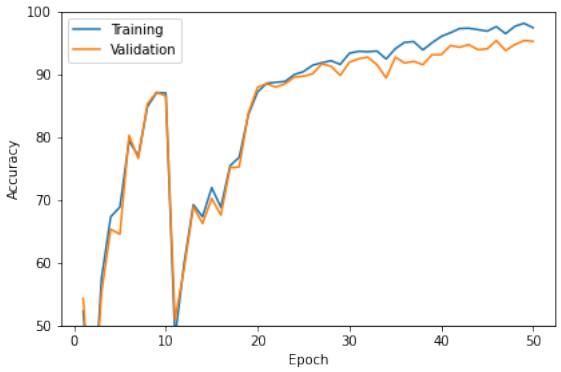

Figure 13. VGG16 Training and Validation Accuracy vs. Epoch.

ResNet34 ends up with a reasonable performance of 92.54% accuracy on the test set. Similar to LeNet5 and our all-convolutional network, it displays a quite strong overfitting. Examined closely, these three methods share one trait in common in our project, which is not having a dropout layer. On the other hand, we have implemented VGG16 and Alexnet with two dropout layers in the fully-connected portion, which may have introduced lack of fairnes.

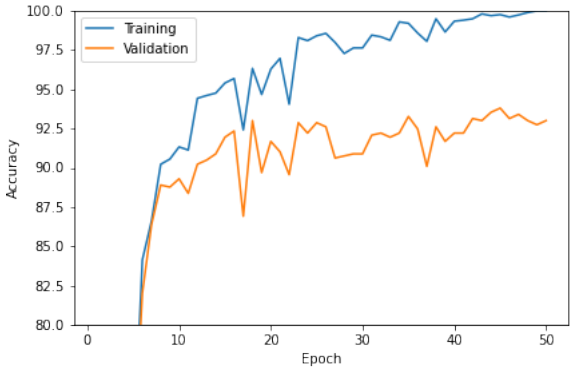

Figure 14. ResNet34 Training and Validation Accuracy vs. Epoch

6. Conclusions

In this project, we have implemented and trained five different common CNN models to classify images as whether people wear masks or not. Compared to previous works that focus on improving one specific method for classification, we are able to visualize different methods implemented with similar parameters and analyze them by comparison. While their performances are all good, it is still clearly visible that AlexNet and VGG16 are outperforming other models, and they are also very different among themselves in terms of model size and training time. VGG-16 has the best performance with the test accuracy of 94.17% although it's much more time-consuming. AlexNet has slightly worse but still robust performance, but it runs much faster compared with VGG-16.

In the Covid 19 context, many locations strictly require visitors to wear face masks, and using an accurate CNN model instead of hiring a person to supervise visitors can save both money and effort. Entities interested in training a model can refer to our paper, contemplate over the advantages and disadvantages of each architecture, and eventually decide on which one to invest in.

One drawback of our project may be that we should not have added dropout layers for AlexNet and VGG16, which might have made the comparison unreliable, as less overfitting is observed in the training curves of these models. Additionally, we have not randomly augmented the training set, which can be a possible cause of severe overfitting. Neither have we computed the mean and standard deviations of each channel. Instead, we have implemented a lazy standardization which assumes that the mean and standard deviation of each channel is 0.5. In order to overcome the problem of overfitting, future studies could also add dropout layers in other models or adjust the batch size to the number before the training and validation accuracy diverge. Besides, they may also randomly rotate or color jitter the images in the training set to encourage the machine to learn only the relevant features.

7. Acknowledgements

While other architectures have their clear definitions, the all-convolutional network is flexible. We want to especially thank our teacher Sebastian Raschka for his design, posted on `https://github.com/rasbt/stat453-deep-learning-ss21/blob/main/L14/3-all-convnet.ipynb`, which we adopted.

In addition, credits shall go to the creators of `https://github.com/pytorch/vision/blob/master/torchvision/models/resnet.py` for their implementation of ResNet34.

8. Contributions

Boyu Shen implemented and trained the LeNet5 and all-convolutional neural network, Zining Tan was responsible for the ResNet34 model, and Ruiting Tong worked on the AlexNet and VGG16 architectures.

In the report, each of us was responsible for writing the CNN architectures that we implemented in the Method

and Results sections. Additionally, Boyu Shen wrote related work, Zining Tan wrote the abstract and introduction, and Ruiting Tong wrote the Experiment section and conclusions.

Zining Tan is responsible for the introduction, motivation and data processing parts of the presentation. Ruiting Tong takes charge of the model architectures and experiment settings. Boyu Shen is responsible for the Results and Discussion, Conclusions parts of the presentation.

References

[1] Deep Learning (Ian J. Goodfellow, Yoshua Bengio and Aaron Courville), MIT Press, 2016

[2] Bushaev, Vitaly. "Stochastic Gradient Descent with Momentum." Medium, Towards Data Science, 5 Dec. 2017, towardsdatascience.com/stochastic-gradient-descent-with-momentum-a84097641a5d.

[3] Springenberg, Jost Tobias, et al. "Striving for simplicity: The all convolutional net." arXiv preprint arXiv:1412.6806 (2014)

[4] Gurav, Omkar. "Face Mask Detection Dataset." Kaggle, 31 July 2020, www.kaggle.com/omkargurav/face-mask-dataset.

[5] Loey, M., Manogaran, G., Taha, M. H. N., & Khalifa, N. E. M. (2021). Fighting against COVID-19: A novel deep learning model based on YOLO-v2 with ResNet-50 for medical face mask detection.Sustainable cities and society, 65, 102600.

[6] LeCun, Y. (2015). LeNet-5, convolutional neural networks. URL: http://yann. lecun. com/exdb/lenet, 20(5), 14.

[7] "LeNet." Wikipedia, Wikimedia Foundation, 16 Mar. 2021, en.wikipedia.org/wiki/LeNet.

[8] Wang, G., & Gong, J. (2019, June). Facial expression recognition based on improved LeNet-5 CNN. In 2019 Chinese Control And Decision Conference (CCDC) (pp. 5655-5660).

[9] Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014)

[10] PyTorch, pytorch.org/.

[11] Matplotlib, matplotlib.org/.