Inference Acceleration for Face Mask Detection

Harshit Monish
University at Buffalo
email:hmonish@buffalo.edu

Hitesh Kumar Balapanuru
University at Buffalo
email:hbalapan@buffalo.edu

1 Abstract

The aim of this project is to find an efficient COVID face mask detection model for Deployment. In deep neural networks the computational cost for inference is higher and is proportional to the number of users/queries. When these deep models are deployed on the cloud, edge devices, mobile phones, etc. for various applications, low latency and less memory consumption are the key aspects for inference to decrease the computational cost on the hardware. In order to reduce the compute demand we can either optimize hardware and software stack or compress the model itself by reducing the number of parameters. Since the latter looks more feasible and in comparison to optimizing the hardware/software stack itself, We aim to explore different model compression techniques in this project.

2 Introduction

2.1 Pruning

Model pruning is the technique of reducing the size of deep learning model by finding small weights in the model and setting them to zero. It helps in speeding up model inference time and also in reducing the model size. Deep learning models typically have large number of weights very close to zero which have minimal contribution to model inference [10]. There are different approaches to prune a model, some of them include:

- Structure: This approach comprise of structured and unstructured techniques [8]. In unstructured technique we remove the weights on a case to case basis whereas in structured approach we remove weights in groups e.g- removing entire channel at a time. Structured pruning typically has better run-time performance characteristics but also has a heavier impact on accuracy of the model. [7]
- Scoring: This approach takes into consideration the threshold value to discard weights . this threshold value is either computed using L1 pruning scores by computing their contribution to the overall tensor vector using taxicab distance or using L2 pruning score computing using Euclidean distance.
- Scheduling: In this approach we add additional steps in the training process to prune the model after few epochs. This is also applied in between fine-tuning steps.

Lottery Ticket Pruning The lottery ticket hypothesis states that for any dense neural network, there exists a random subset within that network that is 10-20% as large, but would converge to the same performance in the same amount of training time as discussed in [4]. This iterative pruning technique produces trained subnets called winning tickets equal to fully parameterized models in performance[9] . The pruning procedure of lottery ticket approach is as follows:

- Initialize the deep neural network with random weights and train it.
- prune the lowest magnitude parameters (L1 norm). Add these to a sparsity mask.
- Reset the model back to scratch i.e rest all the weights back to random.

- Train the model for some number of iterations again. Repeat steps 2, 3 until desired level of sparsity has been reached.

2.2 Knowledege Distillation

Xception Architecture Xception[3] can be said as extreme version of Inception. This architecture leverages two different types of convolutions – Depth wise convolution and pointwise convolution. Depth wise convolution is convolution applied to each channel individually and pointwise convolution is the other name of 1*1 convolution. In this architecture pointwise convolution is followed by depth-wise convolution like inception architecture where n*n convolution is applied after 1*1 convolution.

Separable convolution is the modification to depthwise separable convolution. They are stacked one after other like inception architecture. Residual connections help during the backpropagation stage thus giving better accuracies without residual connection.

Xception outperforms in Top-1 accuracy and Top-5 accuracy compared to VGG-16, ResNet, Inception V3[11] architectures.

Mini Xception Architecture: Inspired by Xception we built a 1 residual connection with 128 size separable convolutions. We further experimented with 128,256 size separable convolution residual blocks. Following are architectures for min Xception.

Figure 1: Left 168 Mini Xception — Right 128,256 Mini Xception

Knowledge Distillation: In simple terms, Knowledge Distillation[6] refers to learning a small model inspired by a large model. We refer to the large model as the teacher and the small model as the student. In Teacher-Student architecture Teacher is pre-trained on a Dataset for a specific task(Example – classification) and student is trained with help of the teacher. During the learning phase, we take a forward pass on Teacher Network and collect logits. In the backpropagation stage, we try to minimize the difference between the logit distribution of Teacher and student. We teach the student to behave or mimic the teacher network for a given set of training images. This is a proven efficient way to train small models on Datasets. In the vanilla Knowledge Distillation (KD) process we use KL Divergence as a distillation loss function with a Temperature parameter of 5,10 and alpha(weighs) set to 0.1. The overall loss is the weighted average of student loss and KL Divergence loss.

3 Implementation

3.1 Pruning

First we create a model with multiple CNN layers. The model architecture comprise of 3 cnn blocks and each block comprise of a CNN layer followed by a batch normalization layer and a Dropout layer to avoid over-fitting. After each CNN block we have a MAX pooling layer as well. Then we train the model for multiple epochs and save the trained model. Next we prune the trained model, we have used one-shot pruning technique on this model.

We begin with applying the unstructured pruning first on our model by iterating through each layer and finding all layers of type Conv2d. Using L1 unstructured pruning to clip 50% of the weight tensor to 0. We then remove the model weights that needs to be pruned from the model. Further we have implemented the L1 structured pruning on the output channels dimensions which removes the lowest scoring channels from the Conv2d layer.

3.2 Knowledge Distillation

Training For the training pipeline, we use transfer learning on Xception architecture initialized with ImageNet weights. We run the finetuning for 20 epochs monitoring validation accuracy with patience 2.

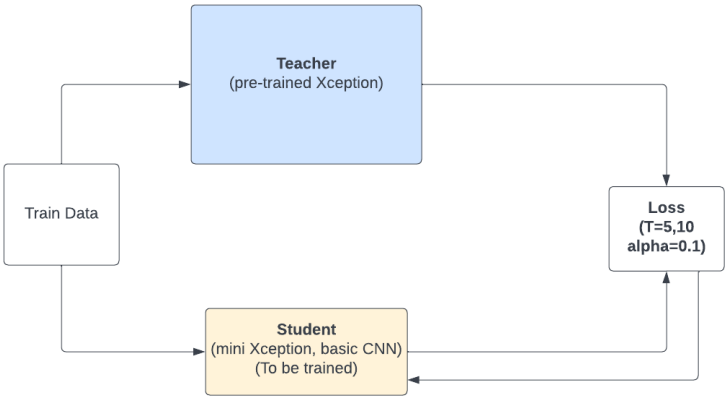

Figure 2: Knowledge Distillation - Student Training

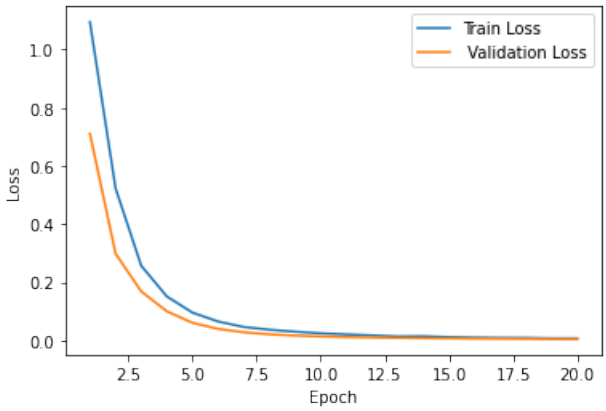

Figure 3: Xception(teacher) Training loss plot

In the second stage, we train Mini Xception, basic CNN models in a teacher- Student setting following the Knowledge Distillation framework. Our goal is to mimic Xception's performance on our student model. **Inference stages:** Once we obtain a trained student model, we make the model sparse in two epochs by finetuning and increasing the sparsity factor from 50 percent to 80 percent. We convert the pruned model is Keras framework to TensorFlow[1] Lite format to get the compressed model.

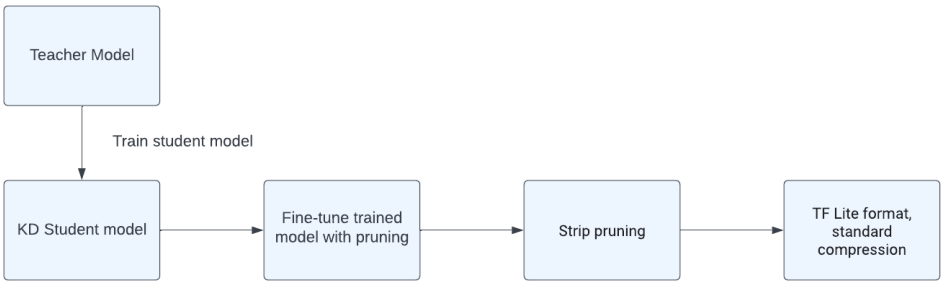

Figure 4: Cascaded Inference

4 Experimental Results

4.1 Dataset

For this project the dataset that we are going to use MaskedFace-Net[2] Dataset. It consists of 137016 quality masked face images each of size 1024x1024. This dataset have three types of masked face images i.e. Correctly Masked Face Dataset (CMFD), Incorrectly Masked Face Dataset (IMFD) and their combination. The labels of this dataset consists of:

- Correctly masked
- Incorrectly masked
  - Uncovered Chin
  - Uncovered nose
  - Uncovered nose and mouth

First the image is classified as correctly masked(CMFD) or Incorrectly masked(IMFD) and then IMFD is further classified as Uncovered chin, Uncovered node and Uncovered nose and mouth. An image is labeled face mask correctly worn if the mask covers the nose, mouth, chin and incorrectly if mask if just covering nose and mouth or mask covering mouth and chin or mask is under the mouth.

4.2 Pruning

We report the loss and accuracy on test data for different pruning ratios. this ratio decides how much percentage of weights needs to me removed from the model. The plots in Figure-6 and Figure-6 shows the effect of pruning on loss and accuracy, higher ratio leads to increase in loss and decrease in the accuracy of the model.The curve for loss for unstructured pruning comes out to be exponential because of the soft cap on model sparsity whereas in the structured pruning has much less fidelity. In case of structured pruning with 5 to 10 percent of the pruning the performance degrades.

Pruning Ratio	Test Loss	Test Accuracy
0.0	0.105	96.2
0.1	0.102	96.17
0.2	0.102	96.17
0.3	0.106	96.12
0.4	0.136	94.57
0.5	0.185	91.49
0.6	0.506	79.14
0.7	1.075	52.47
0.8	1.407	48.39
0.85	1.405	48.26

Table 1: L1 Unstructured pruning results

Pruning Ratio	Test Loss	Test Accuracy
0.0	0.105	96.1
0.1	0.104	96.17
0.2	2.240	48.96
0.3	2.18	48.96
0.4	2.01	48.95
0.5	3.03	48.26
0.6	2.91	48.26
0.7	2.07	48.26
0.8	1.73	50.01
0.85	1.89	50.01

Table 2: L1 Structured pruning results

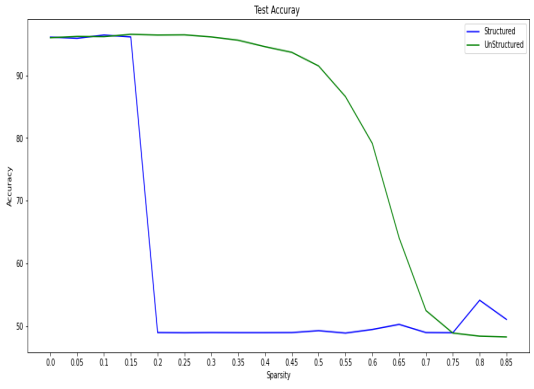

Figure 5: Test Accuracy plot for structured and unstructured pruning

4.3 Knowledge Distillation

For experimentation, We compare training Student model in standalone mode and Student-Teacher Framework. We cascade pruning stage to get final compressed model. We noticed that pruning

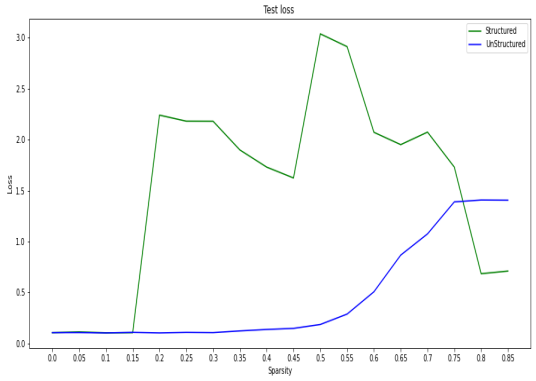

Figure 6: Test Loss plot for structured and unstructured pruning

	Accuracy	**Macro F1**	**Weighted F1**
Transfer learning	0.99	0.99	0.99

Table 3: Xception Model(Teacher)

compresses the model but at the cost of accuracy. Knowledge distillation helps to train a better model.

A slight improvement in metrics can be observed in the case of pruning 128,256 blocks Mini Xception. Upon analysing F1 scores for each class, it was evident due to high imbalance in 2 out of 4 classes.

Knowledge Distillation performs better on all three metrics since it learns imbalanced classes during training phase.

5 Conclusion

In this project we implemented successfully the model compression techniques and reported our results. To achieve a high model compression, we can prune a small model trained in teacher-student framework rather than training from scratch. Knowledge distillation helps to train small student networks by mimicking performance of large pre-trained networks. The code for the project is: https://github.com/harshitmonish/Face-Mask-Detection-with-pruning [5]

Approach	Accuracy	Macro F1	Weighted F1	compression ratio
Teacher- Xception	0.99	0.99	0.99	x
Train from Scratch	0.88	0.73	0.87	363x
Train from Scratch + pruning	0.90	0.68	0.88	1003x
Knowledge Distillation	0.98	0.93	0.98	363x
Knowledge Distillation + pruning	0.90	0.76	0.92	1004x

Table 4: Mini Xception with 128 size block

Approach	Accuracy	Macro F1	Weighted F1	compression ratio
Teacher- Xception	0.99	0.99	0.99	x
Train from Scratch	0.99	0.99	0.99	141x
Train from Scratch + pruning	0.95	0.76	0.93	406x
Knowledge Distillation	0.98	0.94	0.98	141x
Knowledge Distillation + pruning	0.99	0.99	0.99	406x

Table 5: Mini Xception with 128,256 size blocks

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] Adnane Cabani, Karim Hammoudi, Halim Benhabiles, and Mahmoud Melkemi. Maskedface-net–a dataset of correctly/incorrectly masked face images in the context of covid-19. *Smart Health*, 19:100144, 2021.

[3] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017.

[4] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

[5] github. Github, 2020.

[6] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7), 2015.

[7] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip HS Torr. Snip: Single-shot network pruning based on connection sensitivity. *arXiv preprint arXiv:1810.02340*, 2018.

[8] Mingbao Lin, Rongrong Ji, Yuxin Zhang, Baochang Zhang, Yongjian Wu, and Yonghong Tian. Channel pruning via automatic structure search. *arXiv preprint arXiv:2001.08565*, 2020.

[9] Eran Malach, Gilad Yehudai, Shai Shalev-Schwartz, and Ohad Shamir. Proving the lottery ticket hypothesis: Pruning is all you need. In *International Conference on Machine Learning*, pages 6682–6691. PMLR, 2020.

[10] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. Mixed precision training. *arXiv preprint arXiv:1710.03740*, 2017.

[11] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.