Face Mask Recognition

Milestone 2 Deliverable

Team: Emma Ascolese, Stephanie Keene, Brennen Hogan, Ted Donegan

Milestone Accomplishments:

This deliverable includes the results from training the convolutional neural network to classify images in three categories as compared to the two in the previous milestone. Our neural network now includes the capability of determining whether a mask is being worn improperly along with whether it is being worn or not. The results of this milestone demonstrated the clear effectiveness of the three classification models, using both the custom neural network and the resnet34 pre-trained network. The custom neural network had a maximum accuracy of 90.1% after using the topk post-processing function and the pretrained resnet34 had a maximum accuracy of 96.81% after the post-processing function. Overall, the ability to correctly classify faces as no mask worn, mask worn incorrectly, or mask worn correctly proved to be a success in this milestone. Furthermore, we were able to compare the pretrained resnet34 model to a custom neural network that had no previous training. Lastly, we experimented with different learning rates to observe how a higher or lower learning rate impacts each model's accuracy.

Design Decisions:

When approaching the three category classification problem, we had to decide between using a single neural network that could predict one of the three categories or using multiple two classification neural networks that would be layered on top of one other. Ultimately, the decision to use 3 classifications within one trained neural network was made to reduce the computation power and time needed to produce results as training two neural networks would be costly. Furthermore, problematic edge cases would arise when having layered neural networks. We considered having an initial network classify face_with_mask and face_no_mask like in the first milestone. Additionally, we were going to add a second network that would only run on entries labeled face_no_mask, and this network would determine if there was a face_with_mask_incorrect instance. However, if a face_with_mask image was incorrectly classified as face_no_mask, then it was likely that it would be classified as face_with_mask_incorrect by the second neural network. Interdependency between the networks seemed to introduce too many variables and unpredictable behaviors. Furthermore, the accuracy of the second model depends on the accuracy of the first model. By using the three categorization approach, we eliminated the interdependence of the models by allowing all classifications as a possibility with each prediction.

A key difference in a two category and a three category classification problem is the post-processing function. In the first milestone, we used a softmax function to round all results to either 0 or 1. For a three classification problem, we needed to use probabilistic post-processing. One of our first major design decisions for this next step of the project was utilizing the topk function in our post processor. This allowed us to do probabilistic post processing. Within our three classifications (mask, no mask, and mask worn incorrectly), the topk function will select the classification that has the highest probability based on a given input. As we will discuss in the results section below, this decision proved to be an effective method of classification.

The second decision we made was scoring our models based on an effective correct metric. This metric was created due to trends we observed in our model training. The effective correct metric scored an image classification as correct if it was an exact match or if the model correctly predicted 1 or 2 (face_no_mask or face_with_mask_incorrect) for a target that is 1 or 2. We determined that the effectiveness of the model could be accurately represented by this method, since face_no_mask and face_with_mask_incorrect would both be considered “violations” by our model. In a practical sense, a person with either classification would trigger the warning state of the machine.

Similar to milestone 1, we utilized the cross entropy loss function to adjust our weights. Cross-entropy is the default loss function to use for multi-class classification problems (Pytorch.org). In this case, it is intended for use with multi-class classification where the target values are in the set {0, 1, 3, …, n}, where each class is assigned a unique integer value. We also maintained the use of the resnet34 pretrained neural network. This deep neural network proved to be effective in the first milestone, and will be used going forward.

Results and Analysis:

Based on the data in **Tables 1.1**, **1.2**, and **1.3**, the resnet34 pretrained model was proven to be more effective for all learning rates and all success metrics. **Table 1.1** shows that the resnet34 model's average testing loss was 0.015609757 compared to the average testing loss of 0.04231625 for the custom neural network. This demonstrates that on average, the resnet34 model was closer to the target when calculating loss using the Cross Entropy function.

Results in **Table 1.2** and **Table 1.3** are based on the post-processed predictions given by the neural network. For the 3 classification models, we used the topk function to select the class with the highest probability of correctness based on the data. A prediction was counted as correct in **Table 1.2** if the model correctly predicted the class which the image belonged to out of face_with_mask (0), face_no_mask (1), face_with_mask_incorrect (2). In **Table 1.3**, we scored the models based on the effective correct predictions. In this case, we counted a prediction as correct if it was an exact match, or if the model predicted either 1 or 2 for a target that was either 1 or 2. We created this metric due to observed behavior in model training. Oftentimes, the face_with_mask_incorrect (2) category was confused for face_no_mask (1). When considering the application of this technology in a real world setting, both cases 1 and 2 would produce the same result of the machine instructing the user to correct their mask. Comparing **Table 1.2** and **Table 1.3** demonstrates that the model does misclassify categories 1 and 2, but results are oftentimes highly accurate regardless of the metric. Out of a total 1475 possible test entries, the average number of correct predictions for all learning rates was 1266.25 (85.84%) for the custom model and 1420 (96.26%) for the resnet34 network. Similarly, the average number of effective correct predictions for all learning rates was 1301.25 (88.22%) for the custom network and was 1442.25 (97.78%) for the resnet34 model. In both cases, the resnet34 CNN was better than the custom network by about 10%. The custom network saw a greater improvement in accuracy due to the effective correct metric. Both the actual correct predictions and effective correct predictions demonstrate that the resnet34 model was more accurate than the custom network when using the topk post-processing function. The correct prediction percentage for this iteration of the models was slightly lower than the correct prediction percentage for the first two classification models. The data for the initial two classification models can be seen in **Table 2.1** and **Table 2.2**. This decrease in the average was around 3% for the custom model and 1% for the resnet34 model. This is expected behavior for adding an additional classification.

As for the most effective learning rate, the custom network had the lowest average testing loss with the rates of 0.003 and 0.007; however, the learning rate of 0.003 yielded the most accurate results in **Table 1.2** and **Table 1.3**. For the resnet34 model, the learning rates of 0.005 and 0.007 had the lowest average testing loss. In both **Table 1.2** and **Table 1.3**, the learning rate of 0.007 narrowly outperformed the rate of 0.005. Overall, learning rates did not have a dramatic impact on the model's accuracy, but the learning rate of 0.003 seemed to be the best for the custom model and the learning rate of 0.007 produced the best results for resnet34.

The pre-trained resnet34's performance demonstrated the effectiveness of deep neural networks for facial recognition and image data. Furthermore, this model's success demonstrates that prior training, even if it did not include any mask data, could improve the accuracy of a neural network.

Questions/Going Forward:

- Should we be testing learning rates that are more dramatically different than 0.002 increments?
- Could the deeper resnet models be more accurate than resnet34 and further solidify the prediction that deep neural networks are more effective than traditional models
- Should we be looking into changing the custom CNN to use different functions to improve the accuracy?
- Is breaking the model into two separate 2-class predictions more effective than a single 3-class model?

Works Cited

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep Residual Learning for Image Recognition." 25 Dec. 2015. Web. 16 Oct. 2020.

Pytorch.org, "CrossEntropyLoss." Web. 16 Oct. 2020.

Appendix A: 3 Class Data

Average Testing Loss		
Learning Rate	Custom (Non Pretrained)	Resnet34
0.003	0.0383531	0.0148422
0.005	0.0461195	**0.0132650**
0.007	**0.0380162**	0.0135603
0.009	0.0467762	0.0207708
Average	0.04231625	0.015609575

Table 1.1 Average Testing Loss

Correct Predictions (Out of 1475 Possible)		
Learning Rate	Custom (Non Pretrained)	Resnet34
0.003	**1297 (87.93%)**	1420 (96.27%)
0.005	1253 (84.94%)	1423 (96.47%)
0.007	1270 (86.10%)	**1428 (96.81%)**
0.009	1245 (84.41%)	1409 (95.52%)
Average	1266.25 (85.84%)	1420 (96.26%)

Table 1.2 Correct Predictions

Effective Correct Predictions (Out of 1475 Possible)		
Learning Rate	Custom (Non Pretrained)	Resnet34
0.003	**1329 (90.10%)**	1441 (97.69%)
0.005	1292 (87.59%)	1442 (97.76%)
0.007	1306 (88.54%)	**1449 (98.24%)**
0.009	1278 (86.64%)	1437 (97.42%)
Average	1301.25 (88.22%)	1442.25 (97.78%)

Table 1.3 Effective Correct Predictions

Appendix B: 2 Class Data

Average Testing Loss		
Learning Rate	Custom (Non Pretrained)	Resnet34
0.003	**0.004810**	0.008110
0.005	0.022742	**0.00822**
0.007	0.075870	0.004435
0.009	0.027353	0.004481
Average	0.03269375	0.0063115

Table 2.1 Average Testing Loss

Correct Predictions (Out of 1438 Possible)		
Learning Rate	Custom (Non Pretrained)	Resnet34
0.003	**1412 (98.19%)**	1396 (97.08%)
0.005	1296 (90.13%)	1398 (97.22%)
0.007	1092 (75.94%)	**1414 (98.33%)**
0.009	1286 (89.43%)	**1414 (98.33%)**
Average	1271.5 (88.42%)	1405.5 (97.74%)

Table 2.2 Correct Predictions