The Impact of Decreasing Dataset Sizes on Frozen Layer Transfer Learning

By Michael Zheng

Author Bio:

Pengen Zheng, who goes by Michael, is currently a senior at William P. Clements High School located in Sugar Land, Texas. He finds the field of computer vision to be very fascinating and hopes to pursue future studies in machine learning-based computer vision.

Abstract

Transfer learning is a machine learning training method where a model trained for one task gets trained a second time for another task that is usually related yet different from the original task. Applying transfer learning during machine learning training helps to decrease the training time and computational resources needed, as the model has already been pretrained when it was trained for the first task. Transfer learning also helps to compensate for problems like underfitting that might occur when a machine learning model is being trained on a small dataset. This study aims to identify correlations between the effectiveness of transfer learning, specifically frozen layer transfer learning, and the amount of data provided. We accomplish this by applying transfer learning to a machine learning model and feeding it gradually decreasing amounts of data from a data set and observing the model accuracy. We observe that as the amount of provided data decreases, frozen layer transfer learning becomes increasingly less effective.

Keywords: Machine learning, computer vision, transfer learning, frozen layer, YOLOv3, ExDark dataset, computer science, mAP@0.5

Introduction

In recent years, machine learning applications have been increasingly prominent in many different fields. However, with the application of machine learning comes the need to train a machine learning model or multiple models towards a specific goal. During this process, a few problems might arise. For one, depending on the size and complexity of a machine learning model, training such models from the ground up might prove to use a lot of computational resources and time. In some instances, the amount of data in a dataset available for a machine learning model to use for training might be relatively small. If those small datasets are used during ground-up training, it might lead to underfitting, in which a machine learning model is unable to identify dominating trends in a dataset.

A common solution to the aforementioned problems is to apply transfer learning when training a model. Transfer learning is when a model that was originally trained for one task is applied and trained for another different but related task. As the tasks share some similarities, the changes that need to be made to the model when transitioning from the first task to the second task would generally use less computational resources and time compared to training a model for the second task from the ground up. Furthermore, if the amount of training data available for the second task is small, transfer learning helps to compensate for the downfalls in that much of the network would have been set up when the model was trained for the first task, assuming there were large amounts of data available during the process of training the model for the first task. This way, instead of worrying about underfitting, the small amount of data for the second task would act as fine tuners to the model in the transfer learning process. An example of transfer learning might be that a model that was originally trained to identify cars is being re-purposed to identifying trucks, which is a different but related task as there are similar characteristics between a car and a truck.

In our study, we look at a specific transfer learning technique called frozen layers. Frozen layers in transfer learning refer to when a model that has been trained for a task is being trained for a second task, the weights of selected layers won't be changed during the second training process. When layers are frozen, fewer weights need to be modified during backpropagation, which helps to improve the training speed and decrease the computations needed and allows data to make more impact on the unfrozen layers compared to during normal transfer learning. However, given that frozen layers are not updated during the process, this technique may bring decreased accuracy.
In this paper, we will look at how effective frozen layers transfer learning is at compensating for small datasets. We achieve this by training the YOLOv3 Object Detection model (Redmon et al., 2018) using different subsets of the Exclusively Dark (ExDark) Image Dataset (Loh & Chan, 2019). We then measure the final Mean Average Precision of the model after each training as our metrics of evaluation.

Model Training

Algorithm, Dataset, and Training Environment

In this study, we are using YOLOv3, a Real-Time Object Detection system, as our machine learning model. YOLOv3 consists of a Darknet-53 backbone responsible for feature extraction and a YOLOv3 head section responsible for object detection. For our purposes, we are using a PyTorch implementation of YOLOv3 created by Ultralyics (Jocher et al., 2021). In terms of the image dataset, we are using the Exclusively Dark (ExDark) dataset. ExDark is a dataset consisting of 7,363 images in low light environments with 12 different object classes. The ExDark dataset provides enough variability in object classes while maintaining a relatively small number of data. Our models were trained using Google Colab.

Training

We want to accurately observe the effect the size of a dataset would have on task effectiveness. To do so, we first separated the ExDark dataset so that 80 percent of all images are used for training purposes while the other 20 percent are used for validation purposes. The images were separated in a way that

would yield approximately equal distribution of images of different object classes in both the training and validation dataset. After the training images have been separated from the validation images, we then randomly segment the training set into 10 different subsets with equal numbers of images.

To prepare for frozen layer transfer learning, we first train our YOLOv3 model from scratch with the COCO 2017 (Lin et al., 2015) dataset for 300 epochs. COCO 2017 is a dataset with 91 object classes and about 2,118,000 images. Given the large number of images in COCO 2017 and the fact that all object classes in ExDark are a subset of COCO 2017's object classes, this makes COCO 2017 a good candidate as a dataset for the initial training.

After we train our YOLOv3 model from scratch with COCO 2017, we can apply frozen layer transfer learning by first freezing the Darknet-53 backbone, or layers 0 through 10 in the Ultralytics YOLOv3 implementation. This way, we can use transfer learning to further train the parts of YOLOv3 responsible for object detection while leaving the feature extraction as is.

Table 1
Training Configurations and Corresponding Number of Subsets and Epochs

Training configuration	Number of Subsets	Number of Epochs
1	1	1100
2	2	1000
3	3	900
4	4	800
5	5	700
6	6	600
7	7	500
8	8	400
9	9	300
10	10	200

With the Darknet-53 backbone frozen, we train our YOLOv3 model for 10 different training configurations. We first feed our YOLOv3 model 1 subset of the 10 previously segmented ExDark training data and train it for 1100 epochs at 16 images per batch. Then, we reset YOLOv3 to the state immediately after the backbone is frozen and feed it 2 out of the 10 ExDark training subsets and train it for 1000 epochs. These steps are then repeated 8 more times, each time adding one more subset and decreasing the epoch count by 100, until all 10 subsets are being fed into the model, as seen in Table 1. The number of images per batch should remain at 16 images. It should be noted that every new training configuration should have the exact same subsets as before with the exception of the newly added subset. This helps to ensure that for every training configuration except the first configuration, the training data is a strict superset of the previous configuration.

In real-world applications, a machine learning model shouldn't be trained for an extensive amount of epochs as it may lead to the model overfitting. However, for our purposes, we are training each training configuration for more epochs than necessary to extensively observe potential patterns and trends.

Results and Discussion

For our measurement, we will be using specifically mAP@0.5, or the Mean Average Precision for when the IoU threshold is 0.5 and above. In Table 2 we can see the final mAP@0.5 for each training configuration. As expected, as we increase the amount of data provided to the model, the resulting mAP@0.5 increases. Taking a look at Table 3, we find that with increasing amounts of data being provided during training, the increase in final mAP@0.5 shows a generally decreasing trend. This implies that as the amount of data provided decreases, the effectiveness of frozen layer transfer learning becomes more negatively impacted.

Table 2

Amount of Data Provided and the Resulting mAP@0.5

Percentage of Data Provided	Resulting mAP@0.5
0.1	0.1380
0.2	0.1912
0.3	0.2172
0.4	0.2512
0.5	0.2738
0.6	0.2901
0.7	0.3027
0.8	0.3301
0.9	0.3403
1.0	0.3479

Table 3

Final mAP@0.5 Difference Between Each Consecutive Training Configuration

Domain of Percentage of Data Provided	Δ Resulting mAP@0.5
0.1	0.1380
0.2	0.1912
0.3	0.2172
0.4	0.2512
0.5	0.2738
0.6	0.2901
0.7	0.3027
0.8	0.3301
0.9	0.3403
1.0	0.3479

In Figure 1, we graph the progression of mAP@0.5 of each training configuration. Specifically, after each epoch, we calculate the mAP@0.5 by testing each training configuration against the evaluation dataset. We can observe that as the amount of data provided during training decreases, it takes longer for the corresponding configuration's mAP@0.5 to stop fluctuating. The time it takes for a model to stop fluctuating is a reflection of how quickly a model can reach a stage of robustness. From Figure 1, we notice that the less data that's accessible to a model during training, the longer it takes for that model to become robust.

Figure 1

The mAP@0.5 of Each Training configuration During Training.

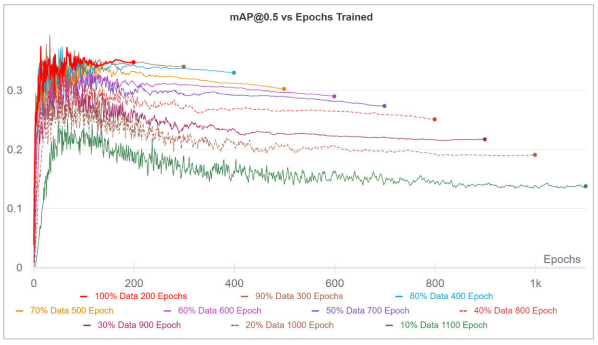

Figure 2

Figure 1 Smoothed Using Running Average Smoothing

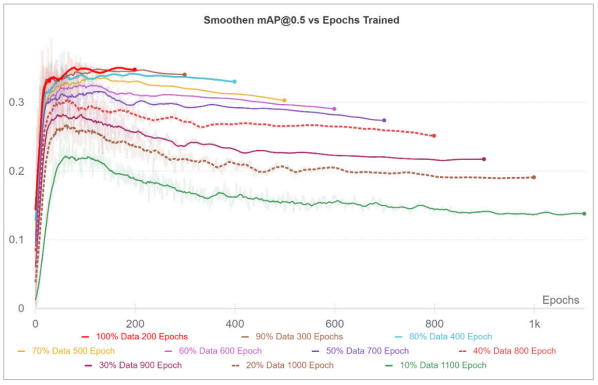

Table 4
Difference Between Highest Final mAP@0.5 after 200 epochs in Figure 2

Percentage of Data Provided	Δ mAP@0.5 Drop Off
0.1	0.0337
0.2	0.0305
0.3	0.0256
0.4	0.0202
0.5	0.0179
0.6	0.0149
0.7	0.0053
0.8	0.0003
0.9	0.0030
1.0	0.0031

As shown in Table 4, as more data is provided to the model during training, the model generally experiences less mAP@0.5 drop-off in the first 200 epochs. We can thus deduce that as less data is provided to a model, the model could overfit easier.

Conclusion

Using the YOLOv3 Object Detection system and the ExDark dataset, we have found that by applying frozen layers transfer learning to machine learning model training, it helps to effectively compensate for minor decreases to the amount of data in a small dataset. However, as more data is removed from a small dataset, frozen layers transfer learning becomes decreasingly effective. Specifically, with decreasing data provided to a model during training, we can determine that frozen layer transfer learning becomes more and more negatively impacted as the model experiences a larger drop-off in mAP@0.5, takes a longer time to reach a stage of robustness, and becomes easier to overfit. In a real-world application, based on time and computational constraints, one could, if needed, reference our results to find a fitting decrease to the transfer learning training dataset that could preserve enough transfer learning effectiveness to produce a final trained model that could meet their expectations.

It should be noted that observations made in this study are based on one specific machine learning model and one specific dataset. In future studies of this project, we plan on implementing the experiment on different combinations of models and datasets outside of YOLOv3 and ExDark to reinforce our conclusion.

References

Jocher, G., Kwon, Y., Guigarfr, perry0418, Veitch-Michaelis, J., Ttayu, Suess, D., Baltacı, F., Bianconi, G., IlyaOvodov, Marc, e96031413, Lee, C., Kendall, D., Falak, Reveriano, F., FuLin, GoogleWiki, Nataprawira, J., … Shead, T. M. (2021, April 12). *Ultralytics/Yolov3: V9.5.0 - YOLOv5 v5.0 RELEASE compatibility update FOR YOLOV3*. Zenodo. Retrieved from https://doi.org/10.5281/zenodo.4681234.

Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C. L., & Dollár, P. (2015, February 21). *Microsoft coco: Common objects in context*. arXiv.org. Retrieved from https://arxiv.org/abs/1405.0312.

Loh, Y. P., & Chan, C. S. (2019). Getting to know low-light images with the exclusively dark dataset. *Computer Vision and Image Understanding*, 178, 30–42. https://doi.org/10.1016/j.cviu.2018.10.010

Redmon, J., & Farhadi, A. (2018, April 8). *Yolov3: An incremental improvement*. arXiv.org. Retrieved from https://arxiv.org/abs/1804.02767.