Aprendizagem Profunda Aplicada – 2022/2023

Assigment 2

Duarte Sousa Cruz
2017264087

Gonçalo Tomaz Arsénio
2017246034

November 25, 2022

1 Introduction

In this assignment we will study the use of transfer learning techniques on deep learning, and also, to exploit an object detector network.

Transfer learning is the improvement of learning in a new task through the transfer of knowledge from a related task that has already been learned [1].

The two components of this assignment are object detection and object classification using transfer learning methods.

2 Part I: Object Classification with Transfer Learning Techniques

2.1 Dataset

For this first part of the assignment we are going to use a custom dataset extracted from the subset of the KITTI object dataset, containing 500 raw RGB images as well as their labels for the 3 object classes (Car, Person, and Cyclist).

A raw RGB image may contain more than one object, where each row in the label file represents an object (bounding box) with the following structure: object class id, normalized center x, normalized center y, normalized width and normalized height.

We want to have a new dataset based on the bounding box image extracted from the previous dataset, to do so we have to convert the provided normalize label to original bounding box coordinates by using the following equations:

$$x1 = w * \left(\frac{x_n - w_n}{2}\right) \qquad y1 = h * \left(\frac{y_n - h_n}{2}\right)$$
$$x2 = w * \left(\frac{x_n + w_n}{2}\right) \qquad y2 = h * \left(\frac{y_n + h_n}{2}\right)$$

being x_n the normalized center x, y_n the normalized center y, w_n normalized width, h_n normalized height, w the width of the image and h the height of the image.

After this we only need to crop the object-centric images, with the bounding box coordinates that we got, and resize those images. In the end we are left with a dataset of 2499 images.

Table 1: Images in Croped Dataset

Total	Car	Person	Cyclist
2499	2074	303	122

Figure 1: Batch example of object-centric dataset

2.1.1 Implementation

Due to the fact that we are using a custom subset of the KITTI dataset, we had to change the last layer to have the capability to identify 3 classes.

We are going to have three different training techniques: train from scratch, train using, as a starting point, a pre-trained model of the the ImageNet dataset, and train by freezing the CNNs' feature extraction parameters.

To do a better and fair assessment of the different training techniques and networks we are going to use the same hyper-parameters for all the combinations.

Table 2: Hyper-parameters used

Training Images	Epochs	Learning Rate	Batch Size	Image Size	Optimizer
1500	20	0.0001	8	128x128	Adam

Pre-trained here means that the Deep Learning architectures have already been trained on a dataset, in our case *ImageNet1k_V1*, a dataset with 1000 classes and millions of images, and therefore carry the resulting weights and biases. When we load a trained model, all parameters default to *.requires_grad=True*, which is fine if we are training from scratch or fine-tuning. However, if we are extracting features and only want to compute gradients for the newly initialized layer, all other parameters should not require gradients. This would prevent the gradients for these parameters from being computed in backward step, which in turn would prevent the optimizer from updating them.

If we do not use transfer learning, the performance of finetuning versus feature extraction depends largely on the dataset, but in general both transfer learning methods provide favorable results in terms of training time and overall accuracy compared to a model trained from scratch. If your dataset is similar to a subset of the ImageNet dataset, this should not matter much. However, if it is different from ImageNet, then freezing will mean a decrease in accuracy. [3] [4]

Table 3: Combination of training techniques

Combination	Pre-train	Freeze CNNs Parameters
1	No	-
2	Yes	Yes
3	Yes	No

2.2 ResNet18

2.2.1 Architecture

Figure 2 shows the architectural layout of the original ResNet-18. This network consists of 17 convolutional layers, a fully linked layer, and an additional softmax layer to perform classification tasks. The convolutional layers use 3×3 kernel filters and the network is designed so that the layers have the same number of filters when the output feature map is the same size. However, the filters in the layers are doubled when the output feature map is halved. Downsampling is performed by convolutional layers with a step size of 2. Finally, average pooling is performed, followed by a fully connected layer with a softmax layer. [2]

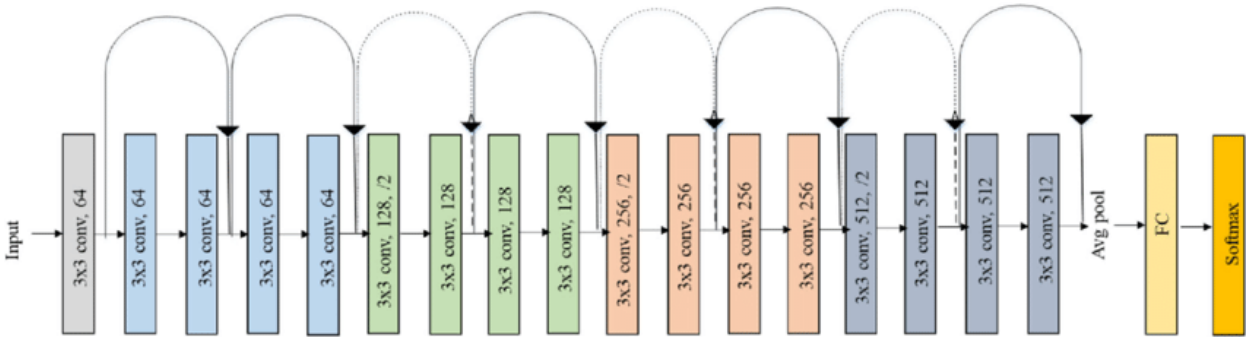


Figure 2: Architecture of ResNet18

2.2.2 Results

Table 4: Evaluation metrics for different training techniques with ResNet18

Combination	mCA (%)	mF1-score (%)	mPrecision (%)	mRecall (%)	Acc Class: Car (%)	Acc Class: Person (%)	Acc Class: Cyclist (%)
1	70.43	70.67	75.00	70.67	96.00	82.00	32.00
2	65.79	75.92	70.00	83.33	99.00	81.00	17.00
3	89.50	92.21	91.67	85.71	99.00	90.00	78.00

By training with the freezing of the CNNs' feature extraction parameters we can see that the performance is worst than when we don't freeze the parameters, because our model will not learn how to extract the necessary features to classify the objects and will only use what he got from the pre-train, so the classification will not be as accurate because the training of the convolutional layers is missing.

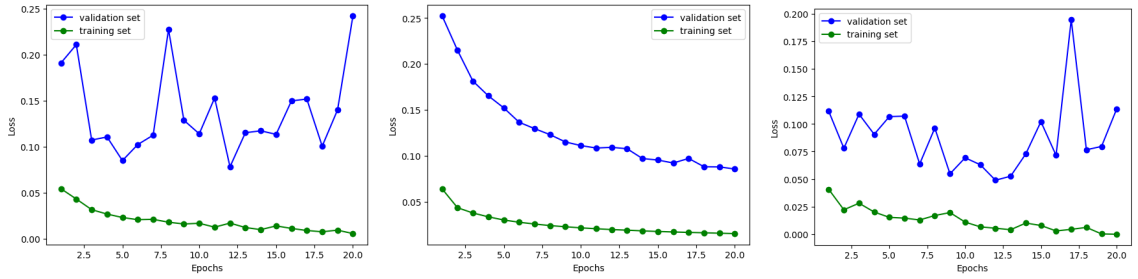
Training from scratch had a slightly better performance, because as we stated above if our dataset is different from ImagNet hen freezing will mean a decrease in accuracy.

As expected if we do not freeze the CNNs parameters, we get an reasonable mCA and we can see that our class accuracy is almost balanced.

Below we have the confusion matrix for the best achieved result, combination 3.

$$\begin{bmatrix} 844 & 6 & 1 \\ 3 & 100 & 8 \\ 1 & 4 & 32 \end{bmatrix} \tag{1}$$

2.2.3 Analysis of Learning Curves



(a) (b) (c)

Figure 3: Plots of the attained loss curves: a) Comb 1 b) Comb 2 c) Comb 3

In figure 3 a) and c) we see that we have a non-representative validation test. This happens when the validation dataset does not provide enough information to evaluate the ability of the model to generalize. As we can see, the training curve looks good, but the validation function moves noisily around the training curve. It could be that the validation data is sparse and not very representative of the training data, so the model has difficulty modeling these examples. [5]

From the figure 3 b) we are dealing with overfitting. This happens when the train loss becomes stable close to zero or the validation loss starts to increase. This means that the model is "overtrained" and loses the ability to generalize and identify classes in new images.

2.3 MobileNetV2

2.3.1 Architecture

The MobileNetV2 network architecture consists of 19 original basic blocks named bottleneck residual blocks. These blocks followed by a 1×1 convolution layer with an average pooling layer. MobileNetV2 builds on the ideas of MobileNetV1 and uses depthwise separable convolution as efficient building blocks. However, V2 introduces two new features to the architecture: 1) linear bottlenecks between layers and 2) shortcut connections between bottlenecks. The intuition behind this is that the bottlenecks encode the intermediate inputs and outputs of the model, while the inner layer encapsulates the model's ability to transform from lower-level concepts, such as pixels, to higher-level descriptors, such as image categories. As with traditional residual connections, the shortcuts allow for faster training and better accuracy. [8]

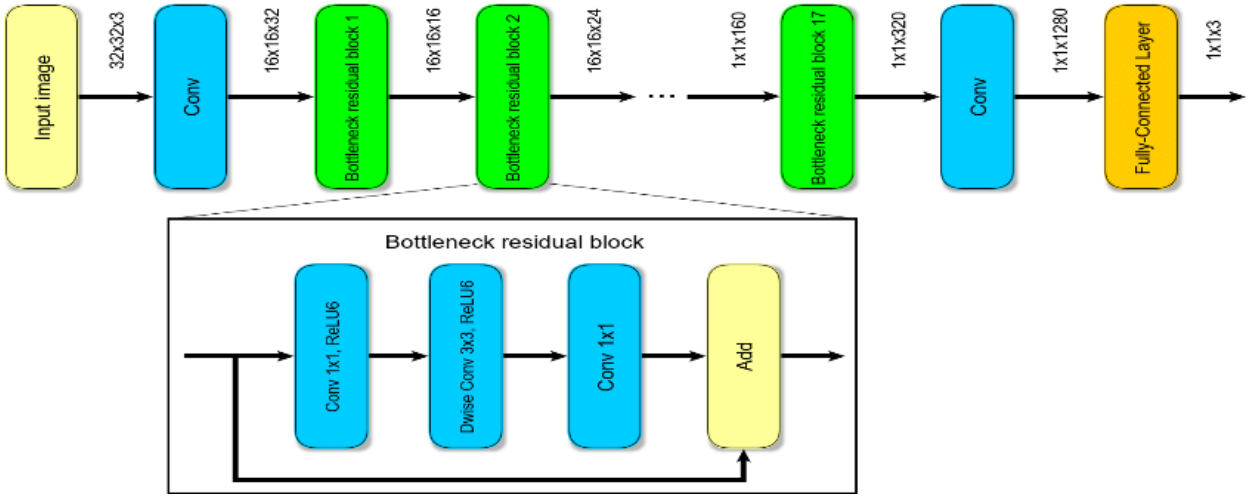


Figure 4: Architecture of MobileNetV2

2.3.2 Results

Table 5: Evaluation metrics for different training techniques with MobileNetV2

Combination	mCA (%)	mF1-score (%)	mPrecision (%)	mRecall (%)	Acc Class: Car (%)	Acc Class: Person (%)	Acc Class: Cyclist (%)
1	49.36	49.00	49.00	49.33	98.00	50.00	0.00
2	66.41	69.67	88.33	66.33	99.00	79.00	20.00
3	71.81	72.67	89.33	72.00	99.00	90.00	25.00

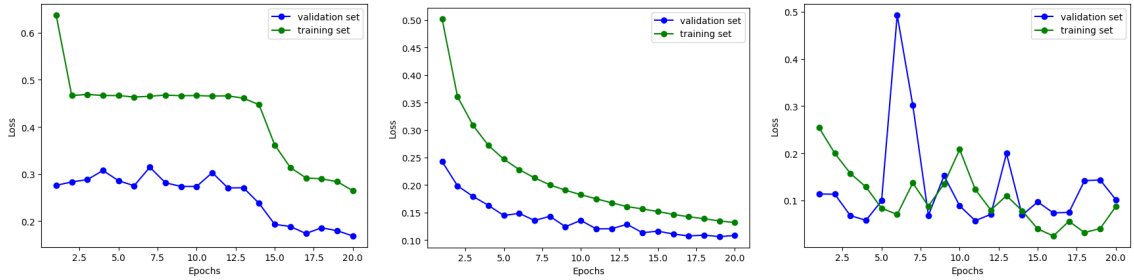
As we can see from the table the transfer learning improved the performance of our model, especially when we didn't freeze feature extraction parameters.

Compared to ResNet18, we can see that MobileNetV2 in combination 3 is more affected by the imbalance of images per class in our dataset.

Below we have the confusion matrix for the best achieved result, combination 3.

$$\begin{bmatrix} 829 & 10 & 4 \\ 4 & 100 & 38 \\ 0 & 0 & 14 \end{bmatrix} \tag{2}$$

2.3.3 Analysis of Learning Curves



(a) (b) (c)

Figure 5: Plots of the attained loss curves: a) Comb 1 b) Comb 2 c) Comb 3

In figure 5 a) we can see an underfit model, the training loss is decreasing and continues to decrease at the end of the plot. This indicates that the model is capable of further learning and possible further improvements and that the training process was halted prematurely.

From the figure 5 b) we are dealing with overfitting. This happens when the train loss has a flat line close to zero or the validation loss starts to increase. This means that the model is "overtrained" and loses the ability to generalize and identify classes in new images.

Finaly, in figure 5 c) we are presented again with a non-representative validation test as in the ResNet18.

2.4 Comparation between ResNet18 and MobileNetV2

By implementing the two networks and analyzing the previous tables and evaluation metrics, we can clearly state that ResNet18 outperforms MobileNetV2. We can also see that ResNet has nearly 12 million trainable parameters, while MobileNet has only about 2 million, because ResNet focuses on computational accuracy, while MobileNet tries to buck the trend of developing deeper and more complicated networks to achieve higher accuracy. However, this comes with a trade-off of size and speed, so MobileNet was developed to solve this problem, i.e., must be able to run on computationally limited platforms.

3 Part II: Object Detection

In this second part of the assignment, we will use the object detector network YOLOv3. We used a subset of KITTI, with 500 RGB images and three classes (car, person and cyclist). We will perform object detection only for the car class, and compare results obtained by training YOLOv3 from scratch and by using transfer learning with the weights of a pre-trained COCO dataset.

3.1 Implementation

YOLOv3 is composed by the Darknet-53 (a network trained on the ImageNet) and Residual networks(ResNet). The network uses 53 convolutional layers (hence the name Darknet-53), with the network constructed from successive 3x3 and 1x1 convolutional layers, followed by a skip connection (introduced by ResNet to allow activations to propagate through deeper layers without decreasing gradient). The 53 layers of the darknet are stacked with another 53 layers for the recognition head, giving YOLO v3 a total of 106 layers of fully convolutional architecture. [9]

3.2 Results

Table 6: Evaluation metrics with different training techniques with YOLOv3

Tranfer Learning	**Epoch**	**Precision (%)**	**Recall (%)**	**AP (%)**	**F1-Score (%)**
Yes	150	67.05	79.46	72.67	72.72
Yes	200	62.31	75.40	63.78	69.23
No	165	44.37	79.23	68.10	56.89
No	200	60.48	72.91	61.15	66.12

If we run the *train.py* file, we get a checkpoint every five epochs that stores the weights from training at that time, so we can use that checkpoint later for object detection and know our evaluation metrics. There is also a file, *YOLO_classID0_mAP_KITTI*, which stores the average precision every five epochs also.

When we use these checkpoints to run *test.py*, we get the evaluation metrics for the epoch of the corresponding checkpoint. In Table 6, we see these metrics for the best and last epoch of each training technique. Analyzing the table, we see that when training YOLOv3 using the provided COCO dataset with a pre-trained model as a starting point, we get the best performance of the model fifteen epochs earlier than when training from scratch, and we might be dealing with some overfitting from that point beyond.

From the information in the table, we can also see that both models have a similar recall, which means that both techniques correctly detect almost the same amount but seeing the precision, we can state that without pre-train the model as way more wrong detection's, getting the worst average precision.

3.2.1 Detection

Now we will demonstrate some examples of the detection's of both techniques for the epochs that got the best average precision.

To do so we add to add some code to *dataset.py* and *detect.py*, so we could make the object detection and save the images with the bounding boxes.

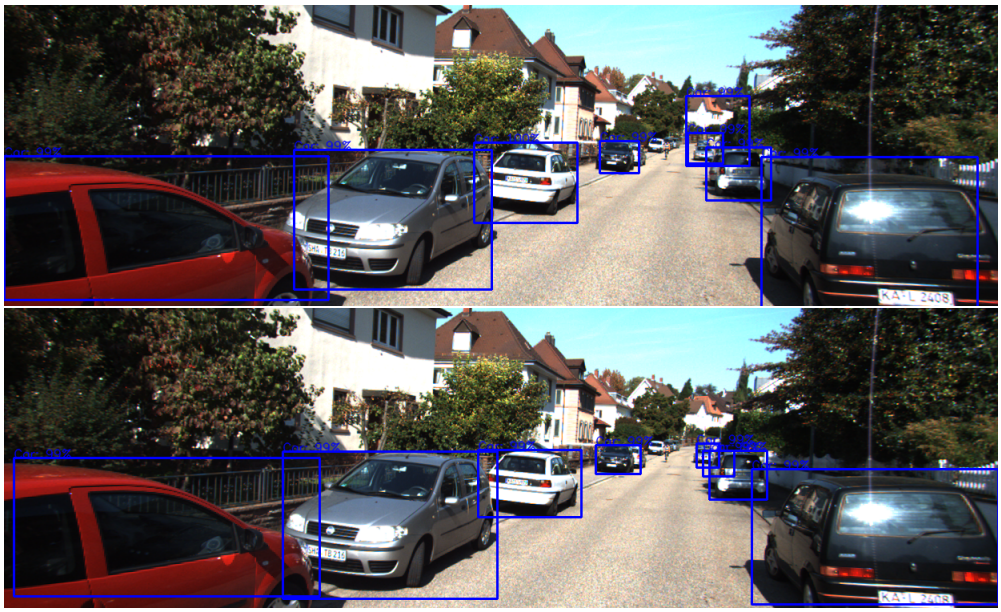


Figure 6: Object Detection: Transfer Learning (above) / Scratch (below)

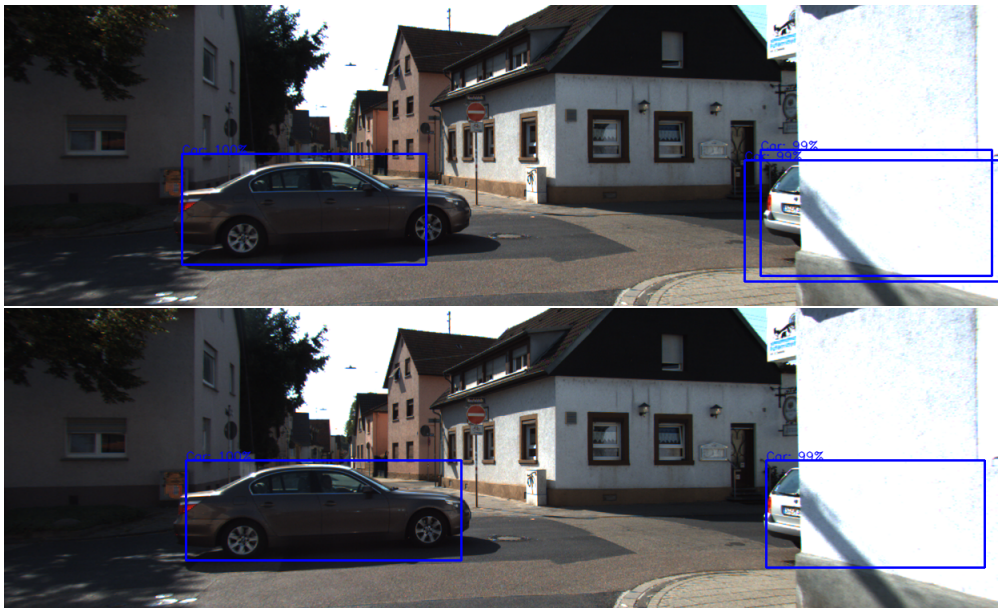


Figure 7: Object Detection: Transfer Learning (above) / Scratch (below)

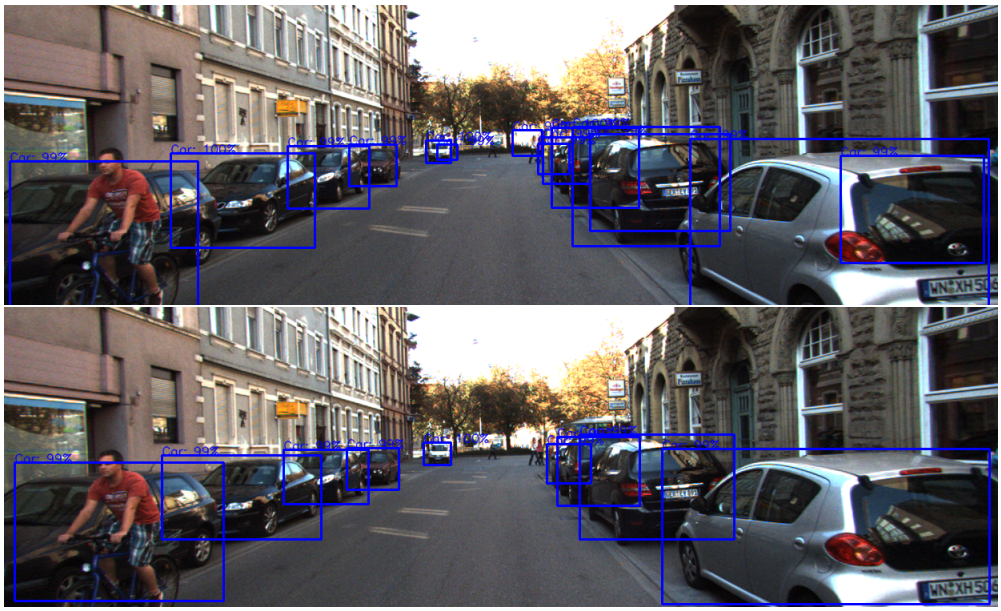


Figure 8: Object Detection: Transfer Learning (above) / Scratch (below)

Figure 9: Object Detection: Transfer Learning (above) / Scratch (below)

From the examples in the figures, we can confirm what we stated above by analyzing the evaluation metrics. Without pre-training, the model has more false detections and sometimes detects the same car more than once. The model trained using the dataset COCO has more accurate bounding boxes, while the model without pre-training detects larger bounding boxes. An interesting thing that we can see in figure 7 is that besides the car from the right is only partially visible the model detects the bounding box making a prediction of the size of the car.

4 Part III: Questions

4.1 Why transfer learning is important in deep learning models?

Transfer learning is the improvement of learning in the new task through the transfer of knowledge from a related task that has already been learned. So it reuses a pre-trained model on a new problem to increase prediction about a new task in transfer learning. For example, is like learn how to ride a motorcycle, already knowing how to ride a bicycle. Even though you are in a different setting, you can transfer the skills, like how to keep your balance or use the brakes, learned from riding a bicycle.

4.2 What are the main differences that you found relevant about the use of transfer learning techniques?

The main differences between traditional machine learning and transfer learning, that we found is that:

- Transfer learning is computationally efficient and helps achieve better results using a small data set;
- Uses knowledge acquired from the pre-trained model to proceed with the task;
- Achieve optimal performance faster than the traditional ML models. It is because the models that leverage knowledge (features, weights, etc.) from previously trained models already understand the features. It makes it faster than training neural networks from scratch.

4.3 Can you further improve the performance of your best CNN architecture in the object classification task?

We could have tried different hyper-parameters and check if it would improve our model.

4.4 Explain the YOLOv3's loss function

The magic of the YoloV3 algorithm comes in the form of its special handcrafted loss function. The calculation of this loss consists of 3 parts:

- The x and y coordinates of the centre point;
- The width and height of the anchor;
- The confidence level;
- If the correct class was predicted

The loss function of YOLOv3 uses different loss functions for the different parameters of the general loss function. Binary Cross-Entropy Loss is used to calculate the cross-entropy of the binary classification tasks. It is used in the general loss function for the parameters of the x and y, confidence factor and for the class classification. For the width and the height is it used the Mean Squared Entropy Loss. Then all these losses are summed and form a general loss for the YoloV3.

4.5 How can you improve the object detection achieved results?

To improve the performance of the YOLOv3 we could have tried Data Augmentation, and changing the batch size or use a different optimizer and see how it would affect the performance.

References

[1] Olivas, Soria Emilio. "Chapter 11: Transfer Learning." Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques, Information Science Reference, Hershey, PA, 2010.

[2] Ramzan, Farheen, et al. "A Deep Learning Approach for Automated Diagnosis and Multi-Class Classification of Alzheimer's Disease Stages Using Resting-State Fmri and Residual Neural Networks." Journal of Medical Systems, vol. 44, no. 2, 2019, https://doi.org/10.1007/s10916-019-1475-2.

[3] Admin. "PyTorch Freeze Layer for Fixed Feature Extractor in Transfer Learning." Knowledge Transfer, 29 Aug. 2021, https://androidkt.com/pytorch-freeze-layer-fixed-feature-extractor-transfer-learning/.

[4] "Finetuning Torchvision Models" Finetuning Torchvision Models - PyTorch Tutorials 1.2.0 Documentation, https://pytorch.org/tutorials/beginner/finetuning_torchvision_models_tutorial.html.

[5] Baeldung. "Learning Curves in Machine Learning." Baeldung on Computer Science, 16 Nov. 2022, https://www.baeldung.com/cs/learning-curve-ml.

[6] Brownlee, Jason. "How to Use Learning Curves to Diagnose Machine Learning Model Performance." MachineLearningMastery.com, 6 Aug. 2019, https://machinelearningmastery.com/learning-curves-for-diagnosing-machine-learning-model-performance/.

[7] Seidaliyeva, Ulzhalgas, et al. "Real-Time and Accurate Drone Detection in a Video with a Static Background." Sensors, vol. 20, no. 14, 2020, p. 3856., https://doi.org/10.3390/s20143856.

[8] Sandler, Mark, et al. "MobileNetV2: Inverted Residuals and Linear Bottlenecks." 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, https://doi.org/10.1109/cvpr.2018.00474.

[9] Digging Deep into Yolo v3 — a Hands-on Guide Part 1. https://towardsdatascience.com/digging-deep-into-yolo-v3-a-hands-on-guide-part-1-78681f2c7e29.

[10] "What Is Transfer Learning? [Examples amp; Newbie-Friendly Guide]." What Is Transfer Learning? [Examples amp; Newbie-Friendly Guide], https://www.v7labs.com/blog/transfer-learning-guide.

[11] "What Is Transfer Learning and Why Does It Matter?" Levity, https://levity.ai/blog/what-is-transfer-learning.