Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Computer Vision Report

Exploring data augmentation for deep learning models in weather image classification

Author: Alberto Formaggio

1 Abstract

Classifying weather images is a crucial task with applications in autonomous driving and intelligent drone autonomy. Human reliance on this task is inconsistent, leading to inaccurate results, and traditional weather forecasts are not real-time, making them unreliable. In this study, we leverage Transfer Learning on Convolutional Neural Networks (CNNs) and a Visual Transformer to develop a robust model applicable to various scenarios, from car windshield images to aerial shots from drones. We explore different data augmentation techniques to enhance model robustness by modifying input data.

2 Data

In this study, we trained our networks on multiple datasets, resulting in a total of 2731 images. The exploited datasets are as follows:

- **Multi-class Weather Dataset (MWD):** This dataset, initially introduced in [1], provides images of diverse weather conditions divided into 4 classes: Cloudy, Rain, Shine, Sunrise. The dataset contains 1125 images.
- **Adverse Conditions Dataset with Correspondences (ACDC) [2]:** This dataset is suitable for training and testing perception methods under adverse visual conditions. The pictures in this dataset were captured with a GoPro camera mounted on the windshield of a car, recording various weather conditions. The classes in this dataset are Clear, Fog, Night, Rain, Snow, comprising a total of 1500 images.
- **UAVid [3]:** The UAVid dataset is a UAV video dataset focusing on urban scenes. The classes in this dataset include Clear, Fog, Night, Rain, with a total of 120 images.
- **Syndrone [4]:** This dataset, not publicly available, provides aerial images similar to the UAVid dataset but in larger quantities. The shared classes between the two datasets are Clear, Fog, Night, Rain, with a total of 1200 images.

To run the files properly, you must adjust the folder structure so that within the **train** folder, there is a subfolder for each class of the dataset. The same organization must be applied to the **test** folder.

3 Setup Description

3.1 Data Processing

Initially, the goal was to create a unified model capable of inference across all datasets, leveraging the diverse sources to gain a broader understanding of the task. However, this approach, while potentially beneficial for the model to acquire general knowledge, posed challenges due to the diversity of the datasets. We initiated training on all datasets, employing a modular implementation that facilitates the straightforward addition or removal of datasets.

Unfortunately, as it can be observed from the aforementioned data description, the diversity of the datasets represented a few challenges to overcome.

3.1.1 Class Imbalance

The MWD dataset comprises classes that are entirely different from those found in other datasets. Additionally, in ACDC, there is a unique class, Snow, not present in the other datasets. Consequently, upon inspecting the images, we arbitrarily defined a mapping between the classes to train our model jointly:

Shine →Clear
Cloudy →Fog

The classes not mentioned kept the same name.

At the end of the mapping, we can find the classes:
[Clear, Fog, Night, Rain, Snow, Sunrise]

The classes Snowy and Sunrise lack any association since they are exclusive to one dataset only. Following this mapping, it becomes evident that a class imbalance will occur, given that classes present in multiple datasets will have a higher appearance rate than those found in only one. To address this, a weight for each class c was defined as:

$$w_c = \frac{1}{|\{(X, y) : (X, y) \in Tr, y = c\}|}$$

where X is the instance and y is the corresponding label. Taking the inverse of the dataset's cardinality ensures that

classes with fewer samples receive higher weights. It's important to note that the weights do not necessarily sum up to 1, and their usage is described in Section 3.1.3.

3.1.2 Dataset Imbalance

Dataset imbalance presented an additional challenge, as all datasets had roughly the same number of images except for the UAVid dataset. Without any adjustment, the model would essentially ignore the images from this dataset, as their contribution to the loss function would be minimal due to their small quantity.

To address this issue, a weight for each dataset d was computed, similar to the approach used for class imbalance:

$$w_d = \frac{1}{|\{X : X \in Tr(d)\}|}$$

. Where $Tr(d)$ is the set of samples belonging to the dataset d in the training set.

3.1.3 Dataset Creation

Once the weights for both the class and the dataset are computed, we can assign to each sample in class c belonging to the dataset d a weight as

$$w = \alpha_c * w_c + \alpha_d * w_d$$

where α_c and α_d are two hyperparameters to be appropriately tuned.

Subsequently, the training set is created by selecting, with replacement, a number of elements equal to the original size of the data. Samples with higher weights have a higher probability of being selected. Since the sampling is done with replacement, samples with higher weights may appear multiple times, while those with very low weights might not even be included in the final training set. The validation and test sets, on the other hand, are left unchanged to reflect real-case scenario usage during inference.

3.2 Data Augmentation

In order to increase the robustness of our model, we defined some augmentation techniques in order to improve the robustness of our model. Inspiration was taken from the work done in [5]:

- **None** (N): No augmentation done.
- **Base** (B): This basic augmentation technique randomly flips the image horizontally and crops the image randomly. Many works in the literature have proven this augmentation technique to be the best performing one in classification even if rather simple.
- **Geometric Simple** (GS): In addition to the Base augmentation, we also do a random rotation of a small quantity (± 20) and do a small adjustment to the brightness (factor 0.1).
- **Geometric Simple + Vertical Flip** (GSV): In addition to the simple geometric transformation, we also flip vertically the image. This is unlikely to work in all the datasets where the sky is largely visible (the sky should be in the topmost part of the image). However, for aerial images, this probably can enhance the results of our prediction.
- **Gaussian** (G): A Gaussian filter of size 3x3 with $\sigma = 1$ is applied to the images in input to blur the edges.
- **Sharpener** (S): The image is sharpened by a factor 0.2 to enhance the edges. This is the opposite of what a Gaussian filter does.

Finally, a normalization of the color channels is performed using the mean and standard deviation of the colors observed in the original training set of the networks pre-trained on the extensive ImageNet dataset.

3.3 Networks

Convolutional Neural Networks (CNNs) play an important role in Computer Vision tasks and, therefore, also in Image Classification. CNNs, thanks to their deep structure, are able to extract very complex features when compared to former approaches. This was until the Visual Transformer ViT made this appearance. There is still debate going on about whether ViTs can fully replace CNNs.

Three architectures were adopted for testing in the project:

- **ResNet50** (RN50) [6]: a key CNN model for computer vision tasks, employs a 50-layer architecture that excels in recognizing and categorizing objects within images. It is known for introducing residual learning, a technique that aids in training very deep neural networks by addressing the vanishing gradient issue.
- **MobileNet** (MN) [7]: a lightweight CNN architecture that, thanks to its design, allows quick inference. This is particularly critical for example in the drone scenario considered, since these devices are usually compact and don't have much space for fitting large computing resources to perform inference.
- **ViT** [8]: By leveraging the self-attention mechanisms, ViT excels in capturing long-range dependencies, with higher efficiency in tasks where spatial relationships play an important role. It's worth noting, however, that ViT benefits from larger datasets.

All three models were pre-trained on the ImageNet dataset and then fine-tuned for the given task.

4 Experiments and Results

4.1 Optimization

: AdamW has consistently demonstrated effectiveness as an optimizer in various scenarios. To streamline the hyperparameter space and capitalize on its proven success, AdamW was selected as the exclusive optimizer. Given its early success, there was no deemed necessity to explore alternative optimizers such as SGD or RMSProp.

To mitigate the risk of overfitting, the weight decay was set to 0.01, preventing the weights from becoming excessively large.

Two high-performing learning rate schedules were evaluated:

- **ReduceLROnPlateau**: This schedule decreases of a multiplying factor of 0.1 the learning rate whenever the evaluation accuracy doesn't increase by 0.05 in the last 3 epochs.
- **CosineAnnealingLR**: Cosine annealing reduces the learning rate using a cosine-based schedule according to the following formula:

$$\eta_t = \eta_{min} + \frac{1}{2}(\eta_{max} - \eta_{min})(1 + cos(\frac{T_{cur}}{T_{max}}\pi)$$

The schedule is reported in Figure 1. In our case, we set $\eta_{min} = 1e-7$ to avoid a non-improving learning rate. The total number of steps is T_max = training batches · epochs and T_cur was increased after the loss for a batch has been back-propagated.

Dealing with a multi-class classification problem, the loss function used was a categorical cross-entropy, defined as:

$$\sum_{i=0}^{BS}\sum_{j=0}^{C} y_{ij} log(p_{ij})$$

where y_{ij} is the ground truth for pattern i belonging to class j, p_{ij} is computed by applying a softmax function to the logits produced by the network, generating thus a probability distribution; BS is the batch size and C is the number of classes of our training set.

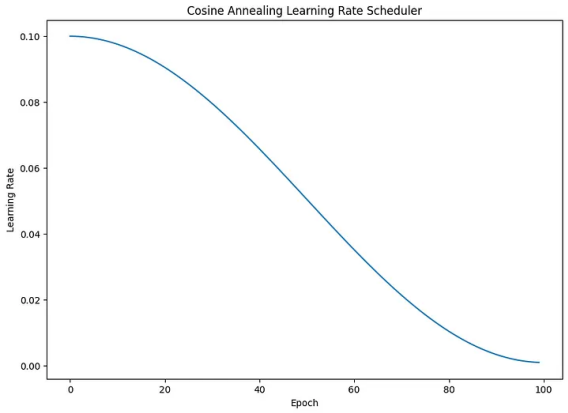


FIGURE 1: Illustration of the Cosine Annealing schedule

4.2 Transfer Learning

In transfer learning for a classification task, a randomly initialized fully connected layer is appended to the end of the pre-trained feature-extractor network. Training the entire network from the initial epoch may yield suboptimal results, as randomly initialized layers may not correctly classify based on the features extracted by the pre-trained network, leading to incorrect weight updates for the entire network. To address this concern, in the conducted experiments, all models underwent initial training for the first 15 epochs, exclusively updating the weights of the fully connected layer. Subsequently, the backpropagation process updated the entire network's weights.

4.3 Model Evaluation

Dealing with a problem with imbalanced classes, as discussed in 3.1, relying solely on the classification accuracy to evaluate our models is not enough. The metrics used are:

- Classification Accuracy: widely used in all the classification tasks, the accuracy simply tells us the number of correctly classified samples.

$$accuracy = \frac{|\{(X,y) \in ds : model(X) = y\}|}{|\{(X,y) \in ds\}|}$$

- Macro Precision: Macro Precision is the average of precision values calculated independently for each class. Precision for a class is the ratio of true positive (TP) predictions for that class to the sum of true positives and false positives (FP). Precision tests how good a model is at making accurate positive predictions.

$$MacroPrecision = \frac{1}{C}\sum_{i=0}^{C}\frac{TP_i}{TP_i + FP_i}$$

- Macro Recall: Macro Recall is the average of recall values calculated independently for each class. Recall for a class is the ratio of true positive (TP) predictions for that class to the sum of true positives and false negatives (FN). Recall assesses how well a model identifies and captures all instances of a particular class, indicating its ability to avoid false negatives.

$$MacroRecall = \frac{1}{C}\sum_{i=0}^{C}\frac{TP_i}{TP_i + FN_i}$$

- Macro F1-score: The Macro F1 Score is a metric used in multiclass classification tasks to assess the overall performance of a model across multiple classes. It calculates the F1 score for each class individually and then takes the average of these F1 scores. The Macro F1 Score provides equal weight to each class, regardless of class distribution. This is to be preferred to other averaging approaches assigning weights to classes since this would make the errors over the minority classes to be negligible, which is not desirable in problems with class imbalances.

$$MacroF1score = \frac{1}{C}\sum_{i=0}^{C}\frac{2 \cdot Precision_i \cdot Recall_i}{Precision_i + Recall_i}$$

It is noteworthy to point out that the training has been carried out on a training set, which constitutes 80% of the samples from the original training set. The remaining 20% of the samples have been used for evaluating the metrics of our model. The validation set was crucial for appropriately tuning hyperparameters to prevent overfitting on the test set. The test metrics were considered only after selecting a model based on the validation dataset.

The results are reported in Table 5.

The parameters used for the models to achieve such results are reported instead in Table 4

We explored also the classes in which some errors during prediction were made, to see if there was some correlation among the models and to determine if we could explain our models' behaviors. The results are reported in Table 3 The training and validation loss for each model are reported in Figure 2.

Metric	Dataset	Networks		
		ResNet50	MobileNet	ViT
Accuracy	MWD	0.978	0.971	**0.982**
	ACDC	0.982	0.956	**0.994**
	UAVid	0.975	0.975	**1.000**
	Syndrone	0.998	0.995	**1.000**
AVG		0.983	0.976	**0.994**
Precision	MWD	0.980	0.972	**0.981**
	ACDC	0.982	0.958	**0.994**
	UAVid	0.977	0.977	**1.000**
	Syndrone	0.998	0.995	**1.000**
AVG		0.984	0.976	**0.994**
Recall	MWD	0.974	0.966	**0.981**
	ACDC	0.982	0.956	**0.994**
	UAVid	0.975	0.975	**1.000**
	Syndrone	0.998	0.995	**1.000**
AVG		0.982	0.973	**0.994**
F1-Score	MWD	0.977	0.969	**0.981**
	ACDC	0.982	0.956	**0.993**
	UAVid	0.975	0.975	**1.000**
	Syndrone	0.997	0.995	**1.000**
AVG		0.978	0.974	**0.994**

TABLE 1: Metrics computation over the test sets of the input datasets

Model	BS	α_d	α_c	η	η_s	ls	Aug
ResNet50	64	2	1	1e-4	cos	0.1	S
MobileNet	16	1	1	5e-5	cos	0.1	B
ViT	16	1	3	5e-5	pl	0.0	B

TABLE 2: Metrics computation over the test sets of the input datasets

Dataset	Classes	ResNet50	MobileNet	ViT
MWD	Cloudy	0	1	2
	Rain	2	3	0
	Shine	4	4	3
	Sunrise	0	0	0
ACDC	Clear	0	2	0
	Fog	0	2	0
	Night	1	1	0
	Rain	5	15	0
	Snow	3	4	4
UAVid	Clear	1	1	0
	Fog	0	0	0
	Night	0	0	0
	Rain	0	0	0
Syndrone	Clear	0	1	0
	Fog	0	1	0
	Night	0	0	0
	Rain	1	0	0

TABLE 3: Errors per class for each dataset

Model	Number of Parameters
ResNet50	23.5 M
MobileNet	**3.2M**
ViT	85.6 M

TABLE 4: Number of parameters for each model, MobileNet has a very low number of parameters, making it useful for applications where computational power is limited.

4.4 Best Model Settings

The results obtained were interesting and need to be subject to some analysis. The pool of parameters tried for each model are the following:

- Batch size BS = [16, 32, 64, 128*]
- α_d = [1, 2, 3]
- α_c = [1, 2, 3]
- Learning rate η = [1e-4, 5e-5]
- Learning rate schedule eta_s = [None, Plateau, Cosine]
- Label smoothing ls = [0, 0.1]
- Augmentation (defined in 3.2) = [B, GS, GSV, G, S]

* The batch size of 128 could only be tested on MobileNet and ResNet50 because it did not fit into the GPU memory during training when using the ViT model.

In general, smaller batch sizes seemed to be better in terms of regularization, leading to better results overall. Also, adding label smoothing led to improved results, as increasing the model's uncertainty about class predictions proved beneficial for CNNs. Other aspects, however, need separate consideration.

4.4.1 ViT

The Visual Transformer turned out to be very robust under a multitude of settings. While increasing regularization for the other two models turned out to be a good choice, neither label smoothing nor augmentation led to better performance. Nevertheless, regardless of the different hyperparameters, the model consistently achieved great results. By training the ViT on all the datasets, we were able to achieve perfect classification on two out of four datasets. Probably the effectiveness on those samples is due to the fact that they both included aerial images, and therefore the two training sets were complementary.

4.4.2 ResNet50

On ResNet50, the results were more spread out. Both small (16) and large (128) batch sizes did not lead to good results, with the performances being worse by more than 20% compared to the case of 32 or 64. Furthermore, geometric transformations, on average, made the results worse by 3%. The results obtained with the Base and Sharpener augmentation were comparable. Using label smoothing made the model better for generalization purposes.

4.4.3 MobileNet

MobileNet, the worst-performing model among those tested, surprisingly achieved great performance when compared to the other two. In fact, the time needed for training was half the time needed for training the ResNet and a third of the time needed for training the ViT. Furthermore, the low complexity allowed for achieving great results. A noteworthy observation is that the training process is quite noisy with significant fluctuations. The evaluation accuracy within a single epoch can vary by more than ±15%. For this reason, it is important to stop training at the right epoch. Setting the class weights higher than the dataset weights led to bad results. Furthermore, batch sizes larger than 16 were worsening the results overall.

It is to be noted that among all the augmentation techniques applied, the None (N) augmentation resulted in perfect classification over the training set but led to slightly worse results than the best techniques. Nevertheless, the difference in the f1-score was less than 1% on average than the augmentation of the best-performing models. This is a bit unexpected since, in general, not doing any augmentation in Machine Learning is likely to lead to overfitting and thus to bad generalization results: what we saw instead is that doing strong or no augmentation is worse than doing simple augmentation techniques like flipping and cropping the image (B) or doing B and making the edges sharper (S).

4.5 Class Errors Explanation

In this section, we will explore the errors made by the models over the test set, as reported in Table 3.

At first glance, it's evident that the class "Snow" of the ACDC dataset seems to be the most problematic, as all the models are performing poorly on this class. Since this class suffers from imbalance, containing samples from only one dataset (while other classes, like "Clear," have been trained on samples from all datasets), one might think this problem is due to the class imbalance not being addressed adequately. If this were the case, one would expect the class "Sunrise" from MWD to have the same problem, but this appears not to be the case.

Upon closer examination of the confusion matrices, (Figure 3) it becomes apparent that the class "Snow" is often confused with "Rain" by ViT. A similar issue occurs with MobileNet, where the class "Rain" is frequently misclassified, often getting confused with "Fog" or "Snow." This confusion is understandable since images belonging to those classes appear similar, even to a human observer.

Regarding the MWD model, a similar pattern emerges: misclassified images belonging to the "Shine" class are often classified as "Sunrise," as these two classes share some characteristics.

4.6 Training on smaller datasets

For completeness, training on a subset of the datasets was also conducted. UAVid and Syndrone share similar features; hence, joint training was performed for all the tested models. The results yielded an accuracy and F1 score of 1.0 for all three models, using the parameters described in Table 4.

To test the robustness of our training on unseen data we tried first training our model on the syndrone dataset and then doing inference over the UAVid dataset. Of course, being the dataset of much smaller size, we reduced the number of epochs and the number of epochs in which only the Linear layer was trained to avoid overfitting. This time, as one can expect by the functioning of neural networks, we got worse results over the test set of UAVid.

Metric	ResNet50	MobileNet	ViT
Accuracy	**0.625**	0.425	0.5
F1-score	**0.592**	0.336	0.335

Table 5: Results of the networks when training on Syndrone and predicting on UAVid

From these results, we can see that ResNet50 was better at generalizing than the other networks. ViT, regardless of its number of parameters, produced a bad generalization when predicting over unseen data.

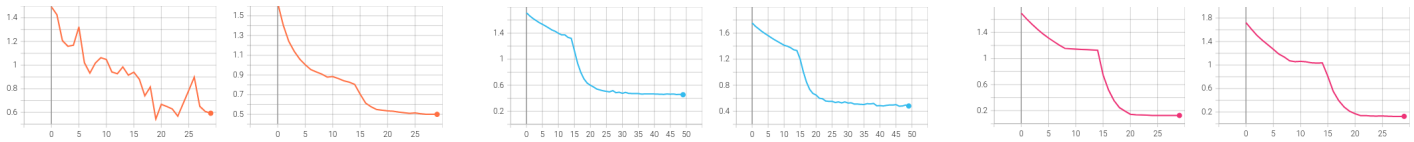
For this reason, we relied on Computer Vision techniques to train a model that was able to generalize better. One of the ideas that came up to our mind was the following: extract the histograms from our image in the LAB and RGB color space using 16 bins, normalize the histograms due to problems that may arise from different image sizes (different size means higher count values) and finally flattenize. The output can then be processed by an MLP and produce an output class. The MLP used was using 2 linear layers and a ReLU activation.

We were able to achieve an accuracy of 0.542 and f1-score of 0.565 by training on syndrone and testing on UAVid. Even though there is still margin for improvement, this simple network, trainable in less than a minute, was able to outperform both ViT and MobileNet. The optimal results were obtained by using SGD with learning rate 0.1, batch size 16, and RGB color space to compute the histograms. LAB in this case was not really stable in training and subject to many fluctuations depending on the parameters chosen. This same model, when tested on syndrone itself, was able to achieve 0.953 accuracy and 0.724 f1-score.

5 Conclusions

In this report, we explored data augmentation techniques for weather image classification using ResNet50, MobileNet, and Visual Transformer (ViT) models. The study addressed challenges like class and dataset imbalance, employing various data processing strategies. ViT consistently performed well, achieving high accuracy across datasets. MobileNet, despite its lightweight nature, showed competitive performance. ResNet50 exhibited more variability.

Overall, due to the competitivity of the results, while relying on a very lightweight network, MobileNet seems to be the best choice for real-world applications in which much computational power is not available. Differently from what one might expect, using regularization or not did not make much difference. Class error analysis highlighted specific challenges, such as confusion between certain weather classes. Overall, this work contributes insights into model effective-

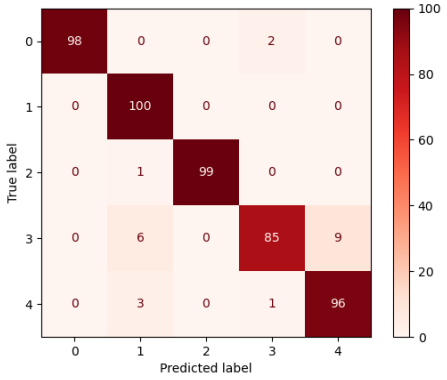


(A) Evaluation and Train Loss for MobileNet

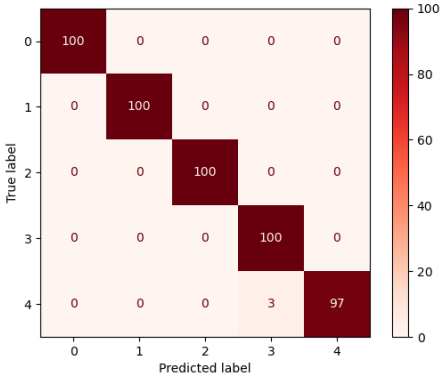
(B) Evaluation and Train Loss for ResNet50

(C) Evaluation and Train Loss for Vision Transformer

FIGURE 2: Evaluation and training loss over different models. In all of them, except for the evaluation loss of the MobileNet, it is possible to observe the epoch in which we started training the full network instead of the classification linear layer only.



(A) Confusion matrix of MobileNet over the ACDC dataset



(B) Confusion matrix of Vision Transformer over the ACDC dataset

FIGURE 3: Confusion matrices for the ACDC datasets for two different models: the classes are, respectively: Clear, Fog, Night, Rain, Snow.

ness and data augmentation impact for weather image classification.

Another trial involved the creation of a simple MLP working over image histogram and thus probably able to generalize better over totally different datasets since it was not related to visual features that the network may observe, but to colors. The results of this simple approach outperformed the powerful models trained on a single dataset and then tested on a new, unseen, one.

Future work may focus on more advanced augmentation techniques, e.g. relying on Deep Convolutional Generative Adversarial networks (DCGANs) to generate weather images artificially.

References

[1] A. Gbeminiyi Oluwafemi and W. Zenghui, "Multi-class weather classification from still image using said ensemble method," in 2019 Southern African Universities Power Engineering Conference/Robotics and Mechatronics/Pattern Recognition Association of South Africa (SAUPEC/RobMech/PRASA), 2019, pp. 135–140.

[2] C. Sakaridis, D. Dai, and L. Van Gool, "ACDC: The adverse conditions dataset with correspondences for semantic driving scene understanding," in Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), October 2021.

[3] L. Madhuanand, F. Nex, and M. Y. Yang, "Self-supervised monocular depth estimation from oblique uav videos," ISPRS Journal of Photogrammetry and Remote Sensing, vol. 176, pp. 1–14, 2021.

[4] G. Rizzoli, F. Barbato, M. Caligiuri, and P. Zanuttigh, "Syndrone – multi-modal uav dataset for urban scenarios," 2023.

[5] S. Yang, W. Xiao, M. Zhang, S. Guo, J. Zhao, and F. Shen, "Image data augmentation for deep learning: A survey," 2023.

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," CoRR, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[7] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," CoRR, vol. abs/1704.04861, 2017. [Online]. Available: http://arxiv.org/abs/1704.04861

[8] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," CoRR, vol. abs/2010.11929, 2020. [Online]. Available: https://arxiv.org/abs/2010.11929