Improving CNN Occlusion Models with Feature Engineering and Normalization

Abish Pius, *Deep Learning Improvements, IEEE,*

***Abstract*—Over the past two decades, convolutional neural networks (CNNs) have become a well established staple in the field of image classification. Even small improvements in training time or accuracy can have great impact in the field, especially if the methodology used can be translated universally across situations. Current research to improve CNN performance is honed in on adjusting the model architecture or adding more layers, altering training methodology, and adding attention mechanisms among other things. The aim of this paper, is to improve a novel tweak on attention mechanisms proposed by *David et al*[2] , termed Occlusion, that results in a generalized performance increase to existing CNN models. Our addition of traditional methods of feature engineering and normalization on top of Occlusion successfully improves the accuracy of the altered CNN architecture as well as decreasing training speed across two standard datasets, Stanford Cars and FGVC-aircrafts. Furthermore, we explore the impact of different transfer learning bases, VGG16 and ResNet50, where we found the smaller netwrok, VGG16, decreases computational time and improves the accuracy of the fit.**

***Index Terms*—Machine Learning, Deep Learning, Stanford Cars, FGVC Aircraft, CNN, Occlusion, Attention Models, Image Classification, Transfer Learning, VGG16, ResNet50.**

I. Introduction

CONVOLUTIONAL Neural Networks (CNNs) are a class of deep neural networks commonly used in image classification, recognition, and segmentation tasks for their ability to extract image patterns that regular connected layers cannot[1]. Though first established in 1980, many advancements in CNN architecture have appeared in the most recent decade due to advancements in computing power[2]. Current research is focused on improving CNN models through deeper architectures, altering training parameters, adding explain-ability, and, of significant interest in this study, implementing attention mechanisms[3].

CNNs, like traditional neural networks, also use neurons with associated activation functions to encode information from the data [4]. The main difference is in how CNNs encode information mainly from spatial dimensions from the data into convolution, pooling and fully connected layers [5]. The input enters a convolutional layer where the output of neurons are connected to local regions of the input through the calculation of the scalar product between their weights and the region connected to the input volume [6]. The pooling layer perform down sampling of the given input reducing the number parameters to fit [7]. Finally, the fully-connected layers will produce the class scores from the activation's that are used for classification [8]. With these three structures, CNNs can take complex architectures to fit general purpose classification. A bulk of these architectures can be used in other model related tasks where only a portion of the architecture needs to be retrained, a method known as Transfer Learning (TL) [9].

Transfer learning was designed as a method by which to leverage knowledge from another task, mimicking how human's learn key features from one task to apply to a different task [10]. In the case of image classification, a great deal of work has been completed across various tasks, including the design of deep architectures for large scale classification. With TL, it is possible to leverage the base architecture of a model used for these large classification tasks and only have to re-purpose a few layers to feed in input and return output, saving both training time and improving accuracy in other similar classification tasks [11]. There are many popular TL architectures, including AlexNet, ResNet50 and VGG16 to name a few[12]. In this supplementary project to an existing paper, we used ResNet50 and focused on improving its predictive capability through attention mechanisms.

Attention mechanisms role in image classification is to add more weight to specific regions in the image, usually by isolating this region during training [13]. The idea is that these regions of the image contain the most useful information for the classification task and by utilizing less of the image speed up training time [14]. One interesting attention mechanism, counter to traditional approaches, is occlusion, where regions the model determines are important after early fits are down-weighted for successive training runs [2]. This is believed to increase the model's generalizability and the main focus of this project.

II. Background and Motivation

THIS project builds off the original work of David et. al in *Playing to distraction: towards a robust training of CNN classifiers through visual explanations techniques*. The authors in this original work present a novel and robust training scheme that integrates visual explanation techniques in the learning process. With this approach the authors claim to improve model generalizability in a broad range of image classification tasks.

The use of visual explanation techniques, by the authors, is similar to the addition of attention into the model. They utilize heat maps to identify the regions of the input images that networks look at when making predictions, allowing the data to be interpreted at a glance. These heat maps are also referred to in the literature as sensitivity maps or class activation maps (CAM) [15]. CAMs are a well-knownprocedure for generating class activation maps using global average pooling in CNNs.

They are nothing more than a weighted linear sum of the presence of these visual patterns at different spatial locations [16]. By simply upsampling the class activation map to the size of the input image, they can analyze the most relevant image regions to identify the particular category. However, CAM can only be used with a restricted set of layers and architectures [2].

To overcome this restriction, the author's use gradient-weighted class activation mapping (Grad-CAM). Grad-CAM uses the gradients of any target concept flowing into the final convolutional layer to produce a coarse localization map, highlighting the regions of the image that are relevant for the prediction. Given an image and a class of interest (e.g, tiger or cat) as inputs, Grad-CAM forward propagates the image through the convolutional part of the model and then through task specific computations to obtain a raw score for the category. The gradients are set to 0 for all classes except for the desired class (tiger or cat), which is set to 1. This signal is then backpropagated to the rectified convolutional feature maps of interest, which are combined to compute the coarse Grad-CAM localization that represents where the model looks at to make the corresponding decision [17].

With the Grad-CAM maps generated, the authors were able to extract regions that they wanted to apply occlusion on so that subsequent training would generalize more to the classification task. They successfully demonstrated this on two public data sets, *Stanford Cars* and *FGVC Aircrafts*, as well as on a private real-case scenario classification of egocentric images.

III. Problem and Tested Concept

FOR our work we will build off the novel occlusion based CNN architecture designed by the authors and improve its ability to generalize through standard machine learning best practices. The authors' did not investigate the impact of classical machine learning techniques to a significant degree. For example, using feature transformations, things like sharpen images or rotating them, to improve validation/testing performance. They utilized early stopping for selecting model weights based on insignificant improvement on the validation set on consecutive runs, even this can be improved with callbacks for saving best validation performance weights. Finally, the authors utilized a rather dense InceptionNetV3 base architecture and in terms of computational time this can be sped up with the use of new smaller more efficient base networks.

Our goal in this project is to improve the accuracy and/or computational efficiency of the authors' best performing zero-occlusion model on the two public datasets, Stanford Cars and FGVC-aircrafts. We tested using a variety of standard feature transformations, including flipping images, zooming images and sharpening images. We also trialed more optimal callbacks selecting for best weights, and we also investigated different base transfer learning networks, one more dense and one less dense.

IV. Experiments and Methodology

THE bulk of the experimental methodology can be found in the original work by the authors, *David et al*, in *Playing to distraction: towards a robust training of CNN classifiers through visual explanation techniques* [2]. It will be recapped in this section along with our additional experimental methodology involving feature engineering, adjusting callbacks, and substituting the transfer learning architecture.

The datasets used in this experiment are publically accessible, the Stanford Cars [18], which contains images of cars along with their make and model, and the FGVC-aircraft [19], which contains images of airplanes and their manufacturer as well as model. For the purposes of this classification experiment the model was the target in both datasets. The Stanford cars dataset contains 16,185 images of 196 car models and it is officially split into 8,144 training and 8,041 test images. The FGVC-Aircraft dataset contains 10,000 images of aircraft, with 100 images for each of 100 different aircraft model variants; and it is officially split into 6,667 training and 3,333 test images. For our experiment, the number of categories in each dataset was reduced to 50 inorder to speed up training/experiment times. Thus, the true tested splits were 2,035 training and 2,010 test images for Stanford Cars and 3,334 training and 1,666 test images for FGVC-aircraft.

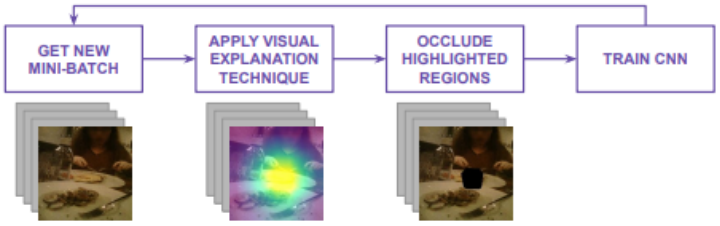

Fig. 1. Experimental workflow taken from *David et al.*'s original work detailing the general model training paradigm.

Figure 1 describes the general CNN training scheme used in the experiment to train the models. The scheme is a modification applied to the traditional batch gradient descent training scheme used by most current models. On each training step, the mini-batch is first fed into the neural network, then the gradient is computed, and finally, the calculated gradient is used to update the weights of the network. We modify the training step to apply the new scheme over each mini-batch with a probability between 0 and 1, else they remain unmodified hence the variability across different full training cycles [2].

Next, using the current weights of the network, we do inference over the current mini-batch and apply a Grad-CAM to get a heat map for each image in the mini-batch. These Grad-CAM heat maps highlight the regions where the current model focuses its attention to classify the corresponding image. After that, we occlude the areas corresponding to those highlighted regions, forcing the model to look at other regions in the image. For each image in the mini-batch, we normalize its heat map and get a weight w between 0 and 1 for each pixel. Next, we select all the pixels whose weight w is over a threshold. The selected pixels are erased by setting them to 0, calling

this approach (Zero) 0-occlusion. Finally, we train our model making use of the occluded images mini-batch [2].

```
Data: trainingSet, model, p, th
Result: the model trained using the proposed approach
for miniBatch in trainingSet do
    r = random(0,1);
    if r ≤ p then
        for (image, label) in miniBatch do
            heatMap = visualExplanation(image, label, model, lastConvLayer);
            heatMap = minMaxNorm(heatMap);
            selectedPixels = [heatMap > th];
            image[selectedPixels] = 0;
        end
    end
    train(model, miniBatch);
end
```

Fig. 2. Algorithm taken from *David et al.*'s original work detailing the general implementation of the Zero(0)-occlusion model.

Figure 2 shows the pseudo-code of the 0-occlusion approach. It is important to notice that once the mini-batch is modified, the training step continues as usual with the gradient calculated and the weights updated. The goal for the model is to not forget what the occluded regions mean, but learns to recognize other parts of the image before making a decision. This is coded as the occluded images are used only for some mini-batches, according to the p hyper-parameter, while the original ones are used for the rest of them. Grad-CAM was selected as it does inference once per image while other visual heatmap techniques do inference several times per image, which makes the former have a significant computational advantage. Again, by occluding the Grad-CAM highlighted relevant regions, the model improves its robustness and generalization capabilities [2].

Moving onto the parameters selected in training the CNN architectures, we used the Adam optimization algorithm with the following parameters: learning rate alpha = 0.001, beta1 = 0.9, beta2 = 0.999, and epsilon = 0.0000007. In the training step, we used a batch size of 64 and the images were resized to 224 × 224. We applied the 0-occlusion with p = 0.25 and the threshold was set to 0.85. We used the weights from the InceptionV3 model pre-trained on ImageNet[20] and then we fine-tuned them using the corresponding dataset and our training scheme. In order to evaluate the performance of the proposed models and make a fair comparison with other approaches, we trained the model and evaluated it on the isolated test set then computed some popular metrics in image classification tasks: accuracy, precision, recall, and F-score (F1). These metrics are defined below[2]: *Note: TP, FP, TN, and FN stand for true positives, false positives, true negatives, and false negatives, respectively

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

As for this projects additions to *David et al.*'s work, we propose three modalities: adding feature transformation, adding normalization and callbacks, and utilizing different transfer learning architectures. We incorporated two standard image prepossessing techniques into our models, horizontal flip and image zoom, using the TensorFlow Imagedatagenerator class. These feature transformations were added to the training data set, including when 0-occlusion occurs on a selected mini-batch. Furthermore, to improve model generalizability, early stopping using TensorFlow callbacks was implemented using training loss with patience set at 10 and min delta at 0.001. Callbacks operate by stopping the training early thereby reducing the occurance of overfitting to the training dataset [23]. Finally, we also tried changing the transfer learning algorithm from InceptionV3 to both ResNet50 [21] and VGG16 [22], less dense and more dense networks, respectively. The idea being to enhance either training speed or accuracy.

In addition, we employed extra leniency on experiments so as to finish within reasonable time constraints. We capped the number of classes present in each dataset to 50, adjusting the number of images trained on as required, and we also limited training to within 100 epochs, due to runtime usage restrictions on Google Colab.

V. Results and Analysis

THE authors' of the original work fit their model to the full 196 classes for the Stanford Cars data and the full 100 classes for the FGVC-aircrafts. Their evaluation metric based results for both public datasets are displayed in Figure 3. Across five trials, they had an accuracy for the 0-occlusion model of 0.871 and 0.749 for the Stanford Cars and FGVC-aircrafts datasets respectively, both better than just using the InceptionNetV3 architecture. Likewise, the other classification metrics precision, recall and F-1 score were also best in the 0-occlusion model giving credence to the idea that the model did generalize better post training. Not surprisingly, there is a significant computational trade-off between the convergences of the 0-occlusion model and the base transfer learning model as seen in figure 4. The 0-occlusion model extends nearly 40 epochs and 60 epochs on average over the InceptionNetV3 model on both the Standford Cars and FGVC-aircrafts datasets, respectively. Additionally, in both cases the 0-occlusion model takes 20 seconds longer per epoch on average. This begs to question whether the method is truly better at classification or if it is a secondary benefit of the longer training period.

Interestingly, there is a considerable amount of variability across training cycles as evidenced by the Grad-CAM heatmaps generated by the authors displayed in figure 5. The heatmaps color the weights of the model per pixel with the brightest being those pixels with the most weight in the neural network architecture. If the 0-occlusion is selected for, based

Stanford cars				
	FT-ResNet50	0-occlusion	R-occlusion	1-occlusion
Accuracy	0.849 ± 0.009	$\mathbf{0.871 \pm 0.007}$	0.860 ± 0.009	0.869 ± 0.008
Precision	0.855 ± 0.007	$\mathbf{0.876 \pm 0.007}$	0.866 ± 0.008	0.873 ± 0.008
Recall	0.849 ± 0.009	$\mathbf{0.870 \pm 0.008}$	0.860 ± 0.009	0.868 ± 0.009
F1	0.848 ± 0.009	$\mathbf{0.870 \pm 0.008}$	0.859 ± 0.009	0.867 ± 0.009
FGVC-Aircraft				
	FT-ResNet50	0-occlusion	R-occlusion	1-occlusion
Accuracy	0.731 ± 0.013	$\mathbf{0.749 \pm 0.005}$	0.739 ± 0.012	0.743 ± 0.005
Precision	0.746 ± 0.011	$\mathbf{0.762 \pm 0.005}$	0.755 ± 0.010	0.759 ± 0.004
Recall	0.731 ± 0.013	$\mathbf{0.749 \pm 0.005}$	0.739 ± 0.012	0.743 ± 0.005
F1	0.731 ± 0.014	$\mathbf{0.748 \pm 0.005}$	0.739 ± 0.012	0.743 ± 0.005

Fig. 3. Experimental results for public datasets Stanford Cars and FGVC-aircraft taken from *David et al*'s original work.

on the threshold and probability, these bright yellow spots pixel values are set to 0. In support of the author's original argument, it does seem that this occlusion will force the model to look away from the main body of the image. However, it would be interesting to evaluate whether performance would suffer if the backgrounds were not uniform.

	FT-ResNet50		0-occlusion	
	Standford cars	FGVC-Aircraft	Standford cars	FGVC-Aircraft
Number of epochs	98.8 ± 6.78	113.4 ± 10.97	130 ± 10.38	174 ± 13.87
Seconds per epoch*	153 ± 0.00		175 ± 0.00	

* Network input size: 224 x 224 x 3. Hardware: NVIDIA T4 Tensor Core GPU.

Fig. 4. Computational training times for public datasets Stanford Cars and FGVC-aircraft taken from *David et al*'s original work.

Fig. 5. Grad-CAM heatmaps across different training cycles from *David et al*'s original work. The areas with bright yellow coloring are fully occluded if selected to do (dependent on the probability and threshold parameters) so for that current minibatch.

In our experiments, we reduced the class size from 196 and 100 in Stanford Cars and FGVC-aircrafts, respectively, to 50 classes in order to train the network within Google Colab's resource restrictions. We trialled three changes, adding feature transformations, adding early stopping, and changing the base network. In the end, we settled on utilizing the early stopping with the feature transformations and both of the former in addition to the changing of the base architecture.

We anticipated that the addition of feature transformations and early stopping would improve the ability of the model to generalize as it would not overfit the training dataset. In Figure 6, we did discover this to be the case, with the feature transforms and the early stopping model to outperform the standard model on the testing dataset for both Stanford Cars and FGVC-aircraft on each testing metric. Feature engineering improved the Stanford Cars model accuracy and F1 score, a ratio between precision and recall, nearly 10% .In both cases, the recall was the worst performing metric which indicates our fit has a bias towards guessing correctly as opposed to capturing all the positive cases.

Stanford Cars	Standard	Feature Engineer + Early Stop
Accuracy	0.802	**0.888**
Precision	0.885	**0.935**
Recall	0.728	**0.828**
F1	0.799	**0.878**

FGVC-aircraft	Standard	Feature Engineer + Early Stop
Accuracy	0.0204	**0.948**
Precision	0.0191	**0.9541**
Recall	0.009259	**0.907**
F1	0.0124	**0.929**

Fig. 6. Experiment with feature transformations and early stopping compared to baseline. In both Stanford Cars and FGVC-aircraft the performance metrics increase. Interestingly, the FGVC-aircraft standard model does not converge well for the testing data.

More intriguing, is that the standard model's, those given by *David et. al's* work, learning rate does not converge well for the FGVC-aircraft. In fact, the learning rate was refactored by a magnitude of 10 for our experiment in order to generate the better fit. Looking at the validation loss curves, for Stanford Cars the standard model parameters fit well. However, the validation loss curves for the author's suggested model parameters on FGVC aircraft is atrocious with a final loss after early stopping over 1000. When substituted with the lower learning rate, the loss is much better eventually settling under 1. This suggests that there is some minimum on the loss curve that a higher learning rate is skipping over during the gradient descent train fitting. Another possibility could be that our initialization of weights was bad, however we tested at five different random seeds and observed the same result.

The next experiment was to investigate if a different transfer learning base network would impact the performance metrics of the model. The paper went with InceptionNetV3 which has roughly 8 million weights. We decided to evaluate the impact of using a larger and smaller base network, so we chose VGG16, with 5 million parameters, and ResNet50, with 9 million parameters. Again, for the experimental groups in the FGVC-aircraft data we used the lower learning rate, 0.0001, and we included the feature engineering plus early stopping steps. Here in Figure 8, the larger network, ResNet50, generalized better on the Stanford Cars dataset as observed by all its testing performance metrics. Surprisingly, we have an issue with ResNET50 on the FGVC-aircraft even with the lower learning rate it still fails to find a good generalized fit. The initial thought was that we were stuck in a local maximum

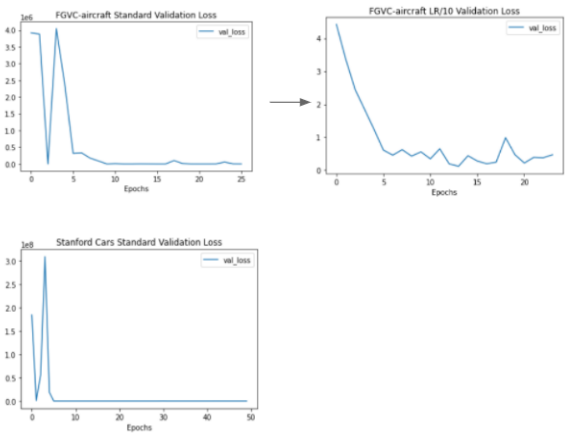

Fig. 7. The validation loss curves for FGVC-aircraft and Stanford Cars. The author's suggested learning rate does not converge well for the FGVC-aircraft with a final loss greater than 1000 (skewed by the axes). It performs as expected with the Stanford Cars dataset.

after training, however even with a larger learning rate and momentum built-in to the adam optimizer, the model still fails to converge. This is definitely a result that should be further evaluated or scrutinized in follow-up projects. Additionally, it may be worthwhile investigating if the VGG16 and ResNet50 base weights were pre-trained on the Standford Cars dataset as the performance is extremely good. Finally, the fact that only 50 classes were fit could also explain why the VGG16 network performed well as it may fail to generalize as well over a larger number of classes with fewer tunable weights.

Stanford Cars	Standard	VGG16	ResNet50
Accuracy	0.802	0.9975	**0.998**
Precision	0.885	0.997	**0.998**
Recall	0.728	0.997	**0.998**
F1	0.799	0.997	**0.998**

FGVC-aircraft	Standard	VGG16	ResNet50
Accuracy	0.0204	**0.972**	0.31
Precision	0.0191	**0.998**	0.104
Recall	0.009259	**0.752**	0.052
F1	0.0124	**0.856**	0.0976

Fig. 8. Performance metrics for experiment switching base architecture from InceptionNetV3 to VGG16 (5million weights) and ResNet50 (9 million weights). For Stanford Cars data the larger ResNet50 has the best performance metrics all around, whereas it seems to have an issue with convergence on the FGVC-aircraft data.

To conclude our experiments, we evaluated how computationally intensive our adjustments to the base 0-occlusion model were in Figure 9. For the Standford Cars dataset the smaller VGG16 architecture is the most computationally efficient training at 23 seconds per epoch, after the initialization (which was not included in the calculation). This is extremely impressive as its fit is quite well with an accuracy of 99.7% and it still trains faster than the InceptionNetV6 even with the additional feature engineering taking place. Also, as expected, the ResNet50 model has the longest training times in both Stanford Cars and FGVC-aircraft. Surprisingly, the feature engineering step on the aircraft dataset has a large computational time investment despite having a lower number of images. The first thought was this was not trained using a GPU, however, we did confirm we trained on a GPU session. The only other thought is that the image size was larger which could explain the over 50 seconds increase in epoch training time over the standard model. Overall, it still seems the addition of feature engineering and shift to VGG16 architecture is the best as it is a good trade-off in terms of increasing model accuracy while reducing training times.

FGVC-aircraft	Standard	Feature Engineer + Early Stop	VGG16	ResNet50
Time per Epoch	48.4	142.1	109.6	136.2

Stanford Cars	Standard	Feature Engineer + Early Stop	VGG16	ResNet50
Time per Epoch	38.2	38.3	23.6	41.4

Fig. 9. Training time per epoch across all experiments. Note: initialization time not included in the calculation.

VI. Conclusion and Future Work

This project built upon the existing work of *David et al* in *Playing to distraction: towards a robust training of CNN classifiers through visual explanation techniques*, improving the performance of their model. We enhanced the 0-occlusion model through feature engineering, early stopping, and replacing the transfer learning base architecture. The idea behind 0-occlusion is to force the model to learn as many features as possible when making a class selection. For this purpose, we apply a visual explanation algorithm to identify the areas on which the model bases its decisions. After identifying those areas, we occluded them and trained the model with a combination of the modified images and the original ones. Therefore the model is not able to base its prediction on the occluded regions and is forced to use other areas.

We trained on a smaller subsample of two public datasets Stanford Cars and FGVC-aircraft due to resource limitations in our colab environment. On both datasets, we confirmed out initial hypotheses that the 0-occlusion model would benefit from early stopping and feature transformations. Surprisingly, we also discovered a potential error in the original proposed model's learning rate for fitting the FGVC-aircrafts, where using their proposed learning rate the model performs atrociously. Furthermore, we discovered that, at least for our smaller sample datasets, the smaller VGG16 TL base architecture generalizes better and trains faster.

According to the author, for future improvements they suggest to apply the same methodology not only to input images but also at different convolutional levels. In other

words, the feature maps obtained at different levels could be analyzed and occluded in the same way that we did with the input images. They believe this idea would force the model to pay attention to different characteristics on the feature maps, thereby improving the robustness of the model at different levels of the learning process. Additionally, they believe incorporating dropout regularization on training loops with occlusion would help enhance generalizability, and possibly be more computationally efficient.

From our experiments, the first thing to implement is to train with our modifications on the full set of classes for both datasets as it is necessary to definitively conclude our observed improvements were actually true. Additionally, along the same line of thought, it would be better to train past 50 epochs and only stop when early stopping kicked in when working with the full set of classes. Next, it would be interesting to unravel why the FGVC-aircraft data does not fit well with a learning rate of 0.001 as well as why the ResNet50 model also performs abysmally. Answering these questions would promote the findings in the project as well drive the path forward to more research.

REFERENCES

1) Albawi, Saad, Tareq Abed Mohammed, and Saad Al-Zawi. "Understanding of a convolutional neural network." *2017 International Conference on Engineering and Technology (ICET).* Ieee, 2017.
2) Kim, Phil. "Convolutional neural network." *MATLAB deep learning*. Apress, Berkeley, CA, 2017. 121-147.
3) Morales, David, Estefania Talavera, and Beatriz Remeseiro. "Playing to distraction: towards a robust training of CNN classifiers through visual explanation techniques." *Neural Computing and Applications* (2021): 1-13.
4) O'Shea, Keiron, and Ryan Nash. "An introduction to convolutional neural networks." *arXiv preprint* arXiv:1511 08458 (2015).
5) Sarıgül, Mehmet, Buse Melis Ozyildirim, and Mutlu Avci. "Differential convolutional neural network." *Neural Networks* 116 (2019): 279-287.
6) Jin, Kyong Hwan, et al. "Deep convolutional neural network for inverse problems in imaging." IEEE *Transactions on Image Processing* 26.9 (2017): 4509-4522.
7) Li, Haoxiang, et al. "A convolutional neural network cascade for face detection." *Proceedings of the IEEE conference on computer vision and pattern recognition.* 2015.
8) Lin, Xiaofan, Cong Zhao, and Wei Pan. "Towards accurate binary convolutional neural network." *arXiv* preprint arXiv:1711.11294 (2017).
9) Tammina, Srikanth. "Transfer learning using vgg-16 with deep convolutional neural network for classifying images." *International Journal of Scientific and Research Publications (IJSRP)* 9.10 (2019): 143-150.
10) Torrey, Lisa, and Jude Shavlik. "Transfer learning." *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques.* IGI global, 2010. 242-264.
11) Ying, Wei, et al. "Transfer learning via learning to transfer." *International conference on machine learning.* PMLR, 2018.
12) Pan, Sinno Jialin, and Qiang Yang. "A survey on transfer learning." *IEEE Transactions on knowledge and data engineering* 22.10 (2009): 1345-1359.
13) Wang, Shiyao, Minlie Huang, and Zhidong Deng. "Densely connected CNN with multi-scale feature attention for text classification." *IJCAI*. 2018.
14) Liu, Fan, et al. "An attention-based hybrid LSTM-CNN model for arrhythmias classification." *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2019.
15) B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, A. Torralba. *Learning deepfeatures for discriminative localization.* IEEE Conference on Computer Vision and Pattern Recognition pp. 2921–2929 (2016)
16) R.R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, D. Batra. "Grad-CAM: Visual explanations from deep networks via gradient-based localization." *IEEE International Conference on Computer Vision pp. 618–626 (2017)*
17) M.D. Zeiler, R. Fergus. "Visualizing and understanding convolutional networks." *European Conference on Computer Vision* pp. 818–833 (2014)
18) J. Krause, M. Stark, J. Deng, L. Fei-Fei. "3D Object Representations for Fine-Grained Categorization." *4th International IEEE Workshop on 3D Representation and Recognition* pp. 554–561 (2013)
19) S. Maji, E. Rahtu, J. Kannala, M. Blaschko, A. Vedaldi. "Fine-Grained Visual Classification of Aircraft." *arXiv preprint* arXiv:1306.5151 (2013)
20) B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, A. Torralba. "Learning deep features for discriminative localization." IEEE *Conference on Computer Vision and Pattern Recognition* pp. 2921–2929 (2016)
21) Alom, Md Zahangir, et al. "The history began from alexnet: A comprehensive survey on deep learning approaches." *arXiv preprint* arXiv:1803.01164 (2018).
22) Qassim, Hussam, Abhishek Verma, and David Feinzimer. "Compressed residual-VGG16 CNN model for big data places image recognition." *2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC).* IEEE, 2018.
23) Wang, Fei, et al. "Backpropagation with callbacks: Foundations for efficient and expressive differentiable programming." *Advances in Neural Information Processing Systems 31* (2018): 10180-10191.