Robust and Generalizable Computer Vision Classification

Ahad Rauf [*1] **Chris Sun** [*1] **Michael Lavva** [*1] **Keisuke Watanabe** [*1]

Abstract

We created a robust computer vision classifier that performs well in a dataset that contains perturbations. To achieve this, we used various data augmentation and other deep learning model techniques, such as model ensembling, image denoising, adversarial training, and attention networks. These methods helped improve the robustness against both naturally perturbed and adversarial datasets. In addition, we created class action map visualizations for our models to help understand how the model makes its classification decisions. Through these techniques, we achieved a 71.1% Top 1 Accuracy and 90.0% Top 5 Accuracy on the Tiny-ImageNet classification challenge.

1. Introduction

Data perturbation is omnipresent in the real world and widely occurs due to compression, resizing, and cropping corruptions in visual input. In spite of the impressive performance on challenging tasks in image classification, the performance of deep networks can be largely affected by data perturbations (Krizhevsky et al., 2017). Therefore, we aim to build an image classifier robust to both natural and adversarial image perturbations. We propose several approaches to stabilizing deep learning neural networks towards making them more robust.

Our main contributions can be summarized as follows:

- We present a well-designed combination of data augmentation methods that improves the validation accuracy (Section 3).
- We generate an adversarial dataset that gives high fooling rates to models not trained for adversarial inputs. We also show how training on this dataset boosts our validation accuracy (Section 3.3).
- We build an ensembled model with ResNet-152D (a custom architecture based off ResNet-152), DenseNet-169, and VGG-19_bn, which outperforms each model individually (Section 4).
- Finally, we try to understand why the model makes a certain classification result and highlight potential adversarial perturbations that could change the prediction using Explainable AI (Section 6).

2. Approach and Tools

All code was written in PyTorch 1.4.0 and Python. The model was trained using Google Cloud's Deep Learning VM instance and both Nvidia Tesla K80 and Nvidia T4.

Our code and final models can be found at the following link: https://github.com/ahadrauf2020/yolo9000.

The model was evaluated on the Tiny ImageNet challenge (https://tiny-imagenet.herokuapp.com/) created by Stanford University. Tiny ImageNet contains 200 classes of 64x64x3 RGB images. There are 500 images per class (100,000 images total) for the training set and 50 images per class (10,000 images total) for the validation set. Validation loss was calculated as the cross-entropy loss between predictions and labels.

We began by preprocessing each image with a denoising filter and applying data augmentation techniques (Section 3). It was then passed to an ensemble of models (Section 4.5), including a derivative of ResNet-152, DenseNet-169, VGGNet-19_bn, and a Residual Attention Network (Wang et al., 2017). We analyzed the average of the outputs for both Top-1 and Top-5 accuracy (Section 4.5). The results are discussed in Section 5. The final output was also visualized using Explainable AI techniques to better understand how the models work (Section 6).

3. Dataset

This section discusses all the data augmentation techniques utilized on the dataset before inputting it to the model.

3.1. Denoising

We found that preprocessing the images using a denoising filter helped improve our accuracy and robustness to noise.

[1]University of California, Berkeley.

Though it can cause a decrease in accuracy for normal training data, it may improve accuracy for adversarial and perturbed images (Rodner et al., 2016). To test the effects of this method, the OpenCV2 function *fastNlMeansDenoisingColored()* was used. Figure 1 shows the effects of denoising on an image. This approach did cause a decrease in training accuracy as expected but it improves robustness on adversarial or perturbed images. Table 1 shows the effects of denoising on accuracy.

(a) Before Denoising (b) After Denoising

Figure 1. The figure shows the results of denoising an image using OpenCV2. Various details are removed from the barrel after denoising in image (b) compared to image (a).

3.2. Data Augmentation

Multiple data augmentation techniques were used when training the model since it's an effective approach in improving robustness (Kim, 2016). A number of data transformations and their effects when applied to the validation dataset of our ResNet152-D model (described in Section 4.1) can be seen in Figure 2. In particular, we found vertical and horizontal flips, grayscaling, random perspective transforms, and random erasures of part of the image to be the most important data augmentations for training networks for robust performance.

3.3. Fast Gradient Sign Method and Universal Adversarial Perturbations

Previous work also shows that training on adversarial examples can improve model robustness (Goswami et al., 2019). In addition to the above data transformations, we explored the Fast Gradient Sign Method (FGSM) (Goodfellow et al., 2015) and Universal Adversarial Perturbations (UAV's) (Moosavi-Dezfooli et al., 2016) as methods for producing adversarial examples to train on. FGSM was helpful for both training and validation, producing high fooling rates on models not trained for adversarial perturbations (Table 1). While not as imperceptible as attacks like Universal Adversarial Perturbations below, with perturbation factor ϵ = 5e-3, this method helped train images to become robust to noise beyond the scope of the denoising filter. A visualization of how the FGSM perturbs images can be found in Figure 3.

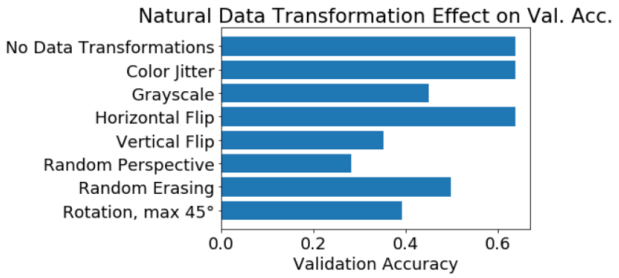


Figure 2. The effect of different data transformations when applied to the validation dataset on a ResNet-152D model trained for 53 epochs. We applied the worst-performing transformations to the training set for future epochs.

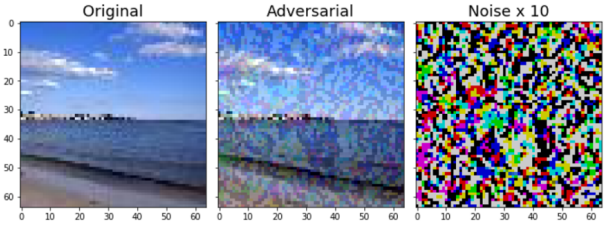


Figure 3. Visualization for how the Finite Gradient Sign Method perturbs a coastline image with ϵ =5e-3. While it isn't a particularly discrete adversarial attack, when used for training it helped provide a robustness beyond just the scope of our local denoising filter.

Table 1. Effects of denoising and Finite Gradient Sign Method adversarial attacks (ϵ =5e-3) on the validation accuracy of a ResNet-152 model, trained for 5 epochs on either a normal or blurred training set.

	Normal Training Set	Blurred Training Set
Normal Validation Set	62.0%	58.8%
Blurred Validation Set	53.0%	59.6%
FGSM Attack on Normal Validation Set	0.68%	0.73%
FGSM Attack on Blurred Validation Set	0.93%	0.89%

Although UAV's gave us very high fooling rates (75.2% on a ResNet-50 model trained for 10 epochs, and after training UAV's for 5 epochs), when training our model on the above data transformations, we noticed no significant improvement in training or validation accuracy. We hypothesize that this is in part due to our processing denoising filter (Section 3.1) and how our model was trained to be resistant to small transformations via data augmentation (Section 3.2).

4. Models

This section discusses all the models trained for a model ensemble. The following models were used: ResNet-152D (a custom variation on the popular ResNet-152 architecture), DenseNet-169, VGG-19_bn, and the Residual Attention Network. It also discusses the results in using model ensembling and snapshot ensembling.

4.1. ResNet-152D

The first model we used was a modified version of the ResNet-152 architecture (He et al., 2015), denoted as ResNet-152D in this paper. Inspired by the robustness of Q-learning, we tried to apply a weight decay factor so the residual connections would pass forward not only the previous layer's output, but also a geometrically scaled version of all previous outputs from the same downsampling group (Figure 5). More concretely, if we denote the ith bottlenecked CONV layer in a downsampling group as $\mathcal{F}_i$, the ith layer's output as y_i (with $i \geq 1$ and y_0 being the output of the previous downsampling group), and the weight decay factor as γ then:

$$y_i = \mathcal{F}_i(y_{i-1}, \{W_i\}) + y_{i-1} + \sum_{n=0}^{i-2} \gamma^{i-1-n} y_n \quad (1)$$

As written, $\gamma = 0$ corresponds to the normal residual operation. After hyperparameter tuning using randomized search with $\gamma \in [0, 0.2]$, $\gamma = 0.045$ gave not only the best validation accuracy but also the fastest training speed (albeit both were relatively minor improvements, especially when using transfer learning from a pretrained model). We hypothesize that, just as residual connections were proposed to speed up early-scale training by feeding forward layers regardless of poorly optimized CONV layer weights, our geometrically scaled residual connections allows the network to remember long-term dependencies instead of just short-term dependencies.

The model was trained for 53 epochs. 18 epochs were on the raw dataset and 35 epochs had data augmentation techniques discussed in Section 3.2 applied. The model achieved a validation accuracy of 65.2%, as shown in Figure 6. In addition to validation accuracy increasing, the decrease in the validation loss after switching datasets shows that the system was able to get to a better local minimum.

4.2. DenseNet-169

An additional model used was DenseNet-169 (Huang et al., 2018). Beginning from the pre-trained model in PyTorch, the model was trained via transfer learning for 50 epochs with a learning rate of 1e-4. It quickly converged to its local optima. After 50 epochs, it achieved a validation accuracy of 67% and training accuracy of 85%, as shown in Figure 6, which shows both the loss and accuracy for training and validation.

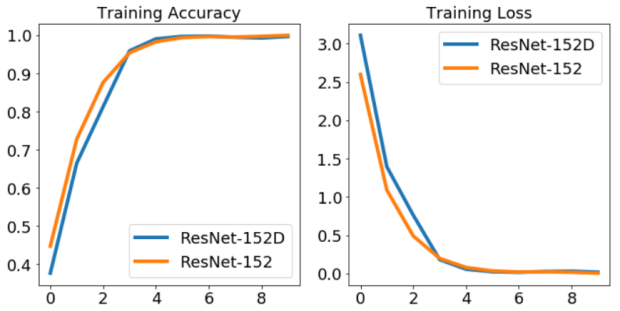


Figure 4. Comparison between ResNet152-D and ResNet-152 for the training accuracy and loss across the first 10 epochs. Despite starting off from a worse initialization point, ResNet152-D trained faster and eventually exceeded the original ResNet152. We found this robustness especially handy when applying data transformations and perturbations, as discussed in Section 3.

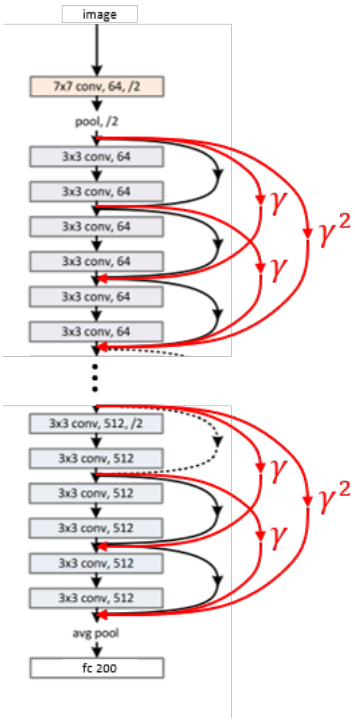


Figure 5. Pictorial diagram showing how ResNet-152D differs from the standard ResNet model. The weight decay γ is applied geometrically across 3x3 CONV layers in the same upsampling group. The upsampling groups represent the size of the upscaled layer input sizes, from 64 at the beginning of the network to 512 at the end of the network.

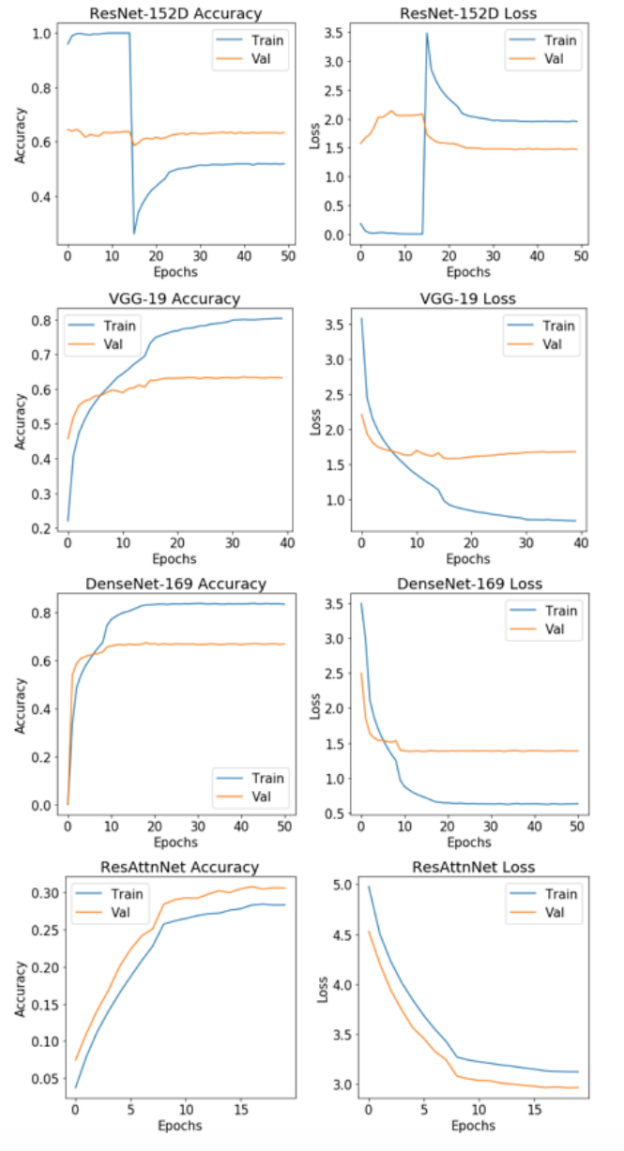


Figure 6. Accuracies and losses for all models. The ResNet-152D graphs show a transition from normal to augmented datasets at epoch 15. Validation accuracy is 63.38% and training accuracy is 51.91%. Validation loss is 1.47 and training loss is 1.95. The DenseNet-169 graphs show that validation accuracy is 66.83% and training accuracy is 83.36%. Validation loss is 1.38 and training loss is 0.63. The VGG-19_bn graphs show that the validation accuracy is 63.23% and training accuracy is 80.35%. Validation loss is 1.68 and training loss is 0.69. The ResAttnNet graphs show that validation accuracy is 30.61% and training accuracy is 28.34%. Validation loss is 2.97 and training loss is 3.12.

4.3. VGG-19_bn

Another model used was VGG-19_bn. Previous work mentioned that VGG-19_bn in particular is a robust model against adversarial perturbations (Roy et al., 2019). Beginning from the pretrained model in the PyTorch library, the model was trained via transfer learning for 40 epochs with a learning rate of 1e-5 and a batch size of 500. It used the data augmentations in Section 3.2 to improve robustness, and a stepped learning rate scheduler (stepLR) that divided the learning rate by 10 every 15 epochs. It achieved a validation accuracy of 63.23% and training accuracy of 80.35%. Figure 6 shows the loss and accuracy for training and validation.

4.4. Residual Attention Networks

Finally, we explored the potential for using residual attention networks. The residual attention network, referred to as ResAttnNet in this paper, is constructed with stacked soft attention modules (Wang et al., 2017). Each Attention Module is divided into two branches: mask branch and trunk branch. The trunk branch is used for feature processing and can be adapted to any network structures. Given the trunk branch output $T(x)$ with input x, the mask branch learns an equivalently sized mask $M(x)$. The output of Attention Module is $H(x) = (M(x) + 1) * T(x)$. The "+1" term embodies the residual attention component, which the paper authors found helped prevent the degradation of attention over long residual chains. The model was trained for 15 epochs and achieved 30.51% validation accuracy.

4.5. Model Ensembling

One technique to improve model robustness is model ensembling (Cheung, 2017). The models explained before, ResNet-152, DenseNet-169, VGG-19_bn, and Residual Attention Network, were used in the model ensemble. We tried various types of ensembling methods, such as majority vote, average of the last layer, and snapshot ensembling (Barnes et al., 2017). Snapshot ensembling and majority vote had much lower validation accuracies than using the average weights of the last fully connected layer, however.

Table 2 shows the validation accuracies of various model ensemble combinations.

5. Results

As a result of using all these approaches discussed before, we were able to improve the accuracy on the dataset up to a Top 1 Accuracy of 71.1% and a Top 5 Accuracy of 90.0% using a model ensemble of ResNet-152D, DenseNet-169, and VGG-19_bn, as shown by Table 2. On a blurred dataset, it achieved a 60.4% This was the highest accuracy that we achieved using the various model ensemble combinations.

Table 2. Top 1 and Top 5 validation accuracy comparisons between the individual models and several ensembles. In the table, ResAttnNet refers to the Residual Attention Network described in Section 4.4. We evaluated our dataset's performance on both a normal dataset and the denoised dataset described in Section 3.1. Note that ensembling by majority vote has no Top 5 Accuracy because the diversity in our model ensembling meant that models rarely agreed on anything outside their top choice.

	Normal Validation Set		Blurred Validation Set
Model	Top 1 Accuracy	Top 5 Accuracy	Top 1 Accuracy
(1) ResNet-152D	63.5%	85.2%	53.1%
(2) DenseNet-169	67.3%	87.2%	49.3%
(3) VGG-19_bn	63.2%	84.3%	57.9%
(4) ResAttnNet	30.5%	58.3%	25.2%
Ensemble (1), (2), (3), (4); Averaging	70.7%	89.7%	60.2%
Ensemble (1), (2), (3); Averaging	**71.1%**	**90.0%**	**60.4%**
Ensemble (1), (2), (4); Averaging	69.7%	89.0%	60.4%
Ensemble (1), (3), (4); Averaging	67.1%	87.5%	55.6%
Ensemble (2), (3), (4); Averaging	67.1%	87.5%	59.2%
Ensemble (1), (2), (3), (4); Majority Vote	67.8%	-	56.5%
Ensemble (1), (2), (3); Majority Vote	68.7%	-	57.2%
Ensemble (1), (2), (4); Majority Vote	62.0%	-	53.5%
Ensemble (1), (3), (4); Majority Vote	63.6%	-	50.0%
Ensemble (2), (3), (4); Majority Vote	63.4%	-	51.9%

6. Extra Credit: Explainable AI

One of the tasks in the project was to include explainable AI, a way to understand why our deep learning model made a certain classification decision. To do this, we created class actions maps (Zhou et al., 2016) for ResNet-152D to show where the model was focusing when it selected its top 3 classes, as shown in Figure 6. The model seemed to focus in particular on the sunglasses' earpieces, the difference between the Christmas stock's red and white regions, and the difference between the refrigerator's top and bottom compartments. While the first two are intuitive, the latter one suggests that the image dataset might have been skewed towards open refrigerators so the model might normally try to cheat by seeing if there's a light gradient between the top and bottom compartments. It's also interesting to look at the image's focus for its 2nd and 3rd top choices. For example, the Christmas stocking is identified as a sock if we ignore the monkey and the color gradient, or instead as an apron if we focus on the area from the ankle region and up.

7. Conclusion

Thus, as a result of this project, we researched and deployed a variety of deep learning techniques to improve robustness in a computer vision problem. With these approaches, we believe that a high accuracy can be achieved, even when adversarial examples are provided in a test dataset.

7.1. Lessons Learned

Several approaches were taken that didn't work as well. First, we tried to use snapshot ensembling but found it to be ineffective compared to other ensembling techniques. Second, we found using the DeepFool algorithm didn't work. There was no improvement in accuracy using the perturbed images for adversarial training.

In order to generate adversarial examples, we trained a Generative Adversarial Network. The generator outputs a perturbation, which is scaled to satisfy a norm constraint. Then we add it to original images, and clip to produce the perturbed images. As a result, we generate an adversarial dataset using image-dependent non-targeted attack. However, the GAN model generates intense noise as shown in Figure 8, so the performance of our model is largely decreased. Additionally, considering that in reality, people usually don't have access to the actual dataset, we decided to give up on the adversarial training method.

In this project, there was a lot of experimentation to try and combine various approaches we found in research. Though certain approaches didn't work as intended, we found interesting insights from them.

7.2. Future Work

In the future, to improve model robustness, we can do further research to find other improvements and do further experimentation. There were many approaches that we could have additionally done. For example, one effective approach is to make a model do pretraining on semi-supervised tasks

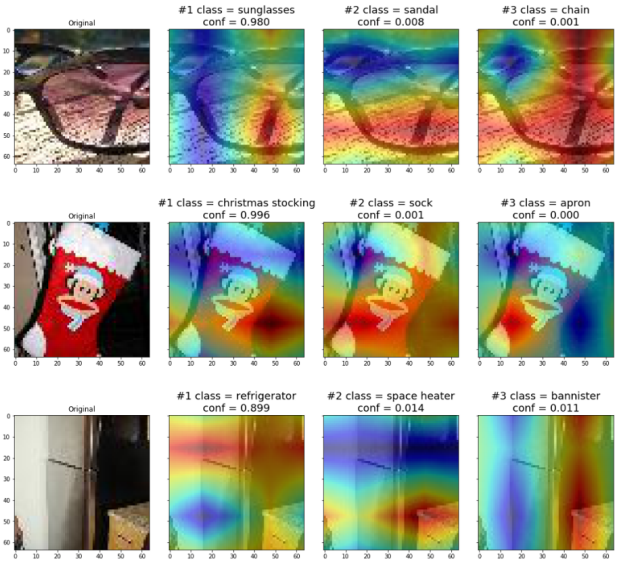
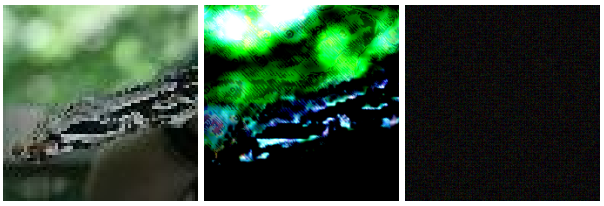


Figure 7. Class action maps for ResNet-152D on sunglasses, a Christmas stocking, and a refrigerator from top to bottom. The class action map above shows what parts of the image the model was focusing on.



(a) Original Image (b) Perturbed Image (c) Noise

Figure 8. (a) The original image. (b) The perturbed image. (c) The noise added to the image.

before classifying images. Examples of such image tasks include filling patches, predicting the amount of rotation, and rearranging a jigsaw (Chen et al., 2020). In addition, while Generative Adversarial Networks failed to train well in our case, more training resources and additional measures against mode collapse might help the generative model produce photo-realistic images.

8. Team Contributions

For this project, each member contributed equally, 25% each. Ahad Rauf worked on adversarial training, explainable AI, and trained the ResNet-152D model. Chris Sun trained the residual attention network and worked on GAN-based adversarial training. Michael Lavva worked on preprocessing techniques and trained the DenseNet-169 model. Keisuke Watanabe worked on snapshot ensembling, model ensembling, and trained the VGG19_bn model.

Each group member also researched papers, contributed in weekly project meetings, and tried various data augmentation techniques and hyperparameter tuning throughout the course of the semester.

Acknowledgements

During this project, we utilized the help of CS182 course staff, techniques from lecture, and other research papers. We especially wanted to thank Roshan Rao and David Chan for their amazing help during office hours in CS182 on exploring state-of-the-art techniques in robust object classification and explainable AI.

References

Barnes, Z., Cipollone, F., and Romero, T. Techniques for Image Classification on Tiny-ImageNet. *cs231n*, 2017. URL http://cs231n.stanford.edu/reports/2017/pdfs/937.pdf.

Chen, T., Liu, S., Chang, S., Cheng, Y., Amini, L., and Wang, Z. Adversarial Robustness: From Semi-Supervised Pretraining to Fine-Tuning. *arXiv: 2003.12862*, 2020. URL https://arxiv.org/abs/2003.12862. arXiv: 2003.12862.

Cheung, V. DenResNet: Ensembling Dense Networks and Residual Networks. *cs231n*, 2017. URL http://cs231n.stanford.edu/reports/2017/pdfs/933.pdf.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. *arXiv:1412.6572 [cs, stat]*, Mar 2015. URL http://arxiv.org/abs/1412.6572. arXiv: 1412.6572.

Goswami, G., Agarwal, A., Ratha, N., Singh, R., and Vatsa, M. Detecting and mitigating adversarial perturbations for robust face recognition. *International Journal of Computer Vision*, 127(6):719–742, Jun 2019. ISSN 1573-1405. doi: 10.1007/s11263-019-01160-w.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. *arXiv:1512.03385 [cs]*, Dec 2015. URL http://arxiv.org/abs/1512.03385. arXiv: 1512.03385.

Huang, G., Liu, Z., van der Maaten, L., and Weinberger, K. Q. Densely connected convolutional networks. *arXiv:1608.06993 [cs]*, Jan 2018. URL http://arxiv.org/abs/1608.06993. arXiv: 1608.06993.

Kim, H. Residual Networks for Tiny ImageNet. *cs231n*, 2016. URL http://cs231n.stanford.edu/reports/2016/pdfs/411_Report.pdf.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, May 2017. ISSN 0001-0782, 1557-7317. doi: 10.1145/3065386.

Langley, P. Crafting papers on machine learning. In Langley, P. (ed.), *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pp. 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.

Moosavi-Dezfooli, S.-M., Fawzi, A., and Frossard, P. Deepfool: a simple and accurate method to fool deep neural networks. *arXiv:1511.04599 [cs]*, Jul 2016. URL http://arxiv.org/abs/1511.04599. arXiv: 1511.04599.

Rodner, E., Simon, M., Robert, F., and Joachim, D. Fine-grained Recognition in the Noisy Wild: Sensitivity Analysis of Convolutional Neural Networks Approaches. *Proceedings of the British Machine Vision Conference*, 2016. URL https://arxiv.org/abs/1610.06756. arXiv: 1610.06756.

Roy, P., Ghosh, S., Bhattacharya, S., and Pal, U. Effects of Degradations on Deep Neural Network Architectures. *arXiv: 1807.10108*, 2019. URL http://arxiv.org/abs/1807.10108. arXiv: 1807.10108.

Wang, F., Jiang, M., Qian, C., Yang, S., Li, C., Zhang, H., Wang, X., and Tang, X. Residual attention network for image classification. *arXiv:1704.06904 [cs]*, Apr 2017. URL http://arxiv.org/abs/1704.06904. arXiv: 1704.06904.

Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., and Torralba, A. Learning Deep Features for Discriminative Localization. *CVPR*, 2016.