# Image Reconstruction Attacks in Distributed DL Systems using Autoencoders

Piyush Nagasubramaniam

May 2, 2023

**Abstract**

Distributed Deep Learning Systems leverage IoT devices and Edge Computing to effectively process data and implement large Deep Learning models at scale. In this work, I present an Image Reconstruction Attack on Distributed Deep Learning Systems for Computer Vision tasks using Autoencoders. In particular, the proposed *black-box* attack targets models that use skip connections and complex model architectures, which makes reconstruction from intermediate results a non-trivial task. I am able to achieve **48.6dB** PSNR (best) and **0.24** MS-SSIM for my reconstructions. I discuss the advantages of incorporating upsampling in a neural network, and how it can improve the reconstruction quality and effectiveness of the attack. As a preventive measure, I posit that using a subset of feature maps during compute-offloading can be an effective mitigation technique at the expense of accuracy and changes in standard model architecture. Code available at: https://github.com/piyushnags/reconstruction

## Contents

# 1 Introduction

In the past decade, there has been a large-scale adoption of IoT devices and computing at the edge. Efficient data collection and diversity in the data are some of the primary motivations for this trend. IoT devices have been able to penetrate the consumer market due to their cost- and energy-effective nature, and in an era that has been dominated by data-driven solutions, efficient large-scale data collection and processing has become increasingly dependent on Edge Computing. However, the lightweight computational capabilities of edge devices makes them unsuitable for processing data at the edge. State-of-the-Art (SOTA) Machine Learning models may have poor execution time on edge devices for real-time data processing requirements, or may not even fit on edge devices depending on the size of the model. The sheer number of IoT devices that are connected to the internet has created new challenges for effectively transporting large volumes of data over to the cloud for data processing.

Consequently, Distributed Deep Learning has become an increasingly important area of research. To address the issue of effective transportation of data, Distributed Deep Learning techniques leverage computational offloading to effectively process data at the edge. Lightweight processing is computed onboard the IoT device whereas computationally intensive tasks are offloaded to the cloud or on-premise centers for processing the data using powerful Deep Learning models.

While distributed architectures have alleviated the bottleneck on data processing, they have also introduced new vulnerabilities. The intermediate results that are sent over a network may contain sensitive information, and can be exploited by adversaries through traditional network attacks like a Man-in-the-Middle (MITM) attack. In the case of videos and images, the intermediate results can be used to reconstruct the original frames (images). The need for privacy-preserving Machine Learning is particularly important when dealing with videos/images that contain information about identity (like faces).

## 1.1 Goals

In this project, I propose a novel image reconstruction attack on Distributed Deep Learning systems for Computer Vision tasks using an Autoencoder. I also posit methods to prevent and mitigate such attacks. In addition to the Deep Learning component, I hope to explore how Image Processing techniques like upsampling can be incorporated into the pipeline for improved image reconstructions.

# 2 Related Work

Many of the adversarial attacks on Deep Learning models for Computer Vision are based on adversarial perturbations in the input. In [1] and [2], the works focus on adding noise that is invisible to the naked eye, but corrupts the feature maps. As a result, classification and object-detection tasks have poor accuracy as the noise gets propagated further down the neural network.

In [3], Oh and Lee explore image reconstruction attacks using intermediate results from deep learning models. However, their work succeeds in reconstructing images for

simpler networks like MobileNetV1 and VGG. They clearly highlight that an image reconstruction attack on deep neural networks and networks that have skip connections is a non-trivial task compared to simpler architectures. The reconstruction accuracy (based on MSE) is 82% for VGG and 8% for Resnet. My work specifically focuses on Deep Learning models that have complex architectures including residual blocks and skip connections. Furthermore, the datasets used in their work do not deal with faces.

In [4], an autoencoder is used to show that image reconstruction attacks are possible on distributed ML systems. However, [4] uses a very simple SqueezeNet model and the size of the dataset used is unclear with respect to the number of parameters in their proposed model. Also, the attack doesn't show results on datasets containing faces.

More concrete comparisons against the baseline methods ([3] and [4]) are presented in the Evaluation section.

# 3 Data

I will be using the CelebA dataset as my primary dataset. CelebA is a collection of celebrity faces with different face poses. There is a good variation in lighting, pose, and background to conduct a study on reconstruction attacks. However, the main subject in all images is a face despite all other variables. The dataset:

1. contains 202,599 color images
2. has no fixed size for the images (resized to 320x320 during training/inference)
3. will not make use of labels

Figure 1 shows a sample of original and normalized images, which will be used as input by Deep Learning models. In the context of this project, the normalized images will be used as input for the encoder. Note that the normalization enhances certain features during feature extraction.

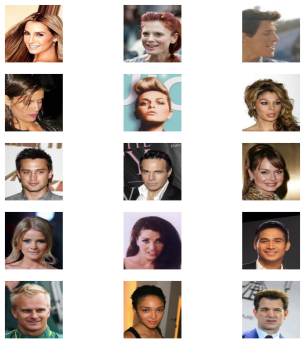
## 3.1 Size Sufficiency

We use 40,000 samples from the dataset in total due to time and resource constraints. However, this is sufficient for the problem addressed in this paper. We can satisfy the size sufficiency if we have roughly 10x as many features as degrees of freedom in the model. In this case, we have $40000 \times 3 \times 320 \times 320 = 12.28B$ features. On average, the decoder has 150k trainable parameters. Clearly, a set of 40,000 samples is sufficient for this model.

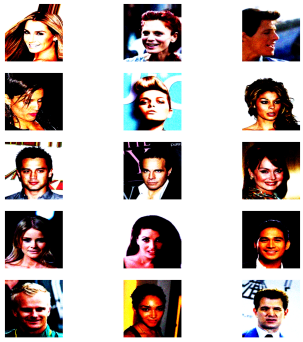
# 4 Methods

The attack will be simulated using an Autoencoder. The encoder will represent a part of the victim model, the latent representation of the autoencoder will correspond to the feature maps sent over the network (sensitive information), and the decoder is the adversarial module that is used to reconstruct the original image.

In Distributed Deep Learning systems, some of the computation is performed on the edge device (usually for bandwidth efficiency). This partial computation is done by the



(a) Original Samples from CelebA



(b) Normalized Samples from CelebA

Figure 1: CelebA Dataset Samples

encoder. The attack presented is a black-box attack, and as such, does not have any information about the underlying architecture of the victim model. We do not make any assumptions about the model weights or parameters. Thus, the design of the decoder is independent of the victim model. It is worth noting that this is a key difference from the work presented in [3], which is a white-box attack.

I will use the MobileNetV3 architecture as the target of the exploit. MobileNetV3 is one of the fastest neural networks for real-time inference and is suitable for inference on edge devices [5]. The systems of interest are those that use computational offloading for real-time data processing. We exploit the vulnerable architecture, which was a tradeoff for improved performance and cost-efficiency. Consider the architecture presented in Figure 2. We use the output of the $4^{th}$ block from the last pooling layer (this block is referred to as the C4 block as per the literature) for the latent representation of the autoencoder. The C4 block is common for almost all downstream tasks like object detection and semantic segmentation, which makes it a suitable input for the adversarial decoder. It is important to note that the encoder is frozen throughout the training process since we do not have control over the victim model's weights. Thus, the quality of the reconstruction is entirely dependent on the training method and the architecture of the decoder.

For the decoder, we test two architectures: *decoder with transposed convolutions* and

| Input | Operator | exp size | #*out* | SE | NL | *s* |
|---|---|---|---|---|---|---|
| $224^2 \times 3$ | conv2d | - | 16 | - | HS | 2 |
| $112^2 \times 16$ | bneck, 3x3 | 16 | 16 | - | RE | 1 |
| $112^2 \times 16$ | bneck, 3x3 | 64 | 24 | - | RE | 2 |
| $56^2 \times 24$ | bneck, 3x3 | 72 | 24 | - | RE | 1 |
| $56^2 \times 24$ | bneck, 5x5 | 72 | 40 | ✓ | RE | 2 |
| $28^2 \times 40$ | bneck, 5x5 | 120 | 40 | ✓ | RE | 1 |
| $28^2 \times 40$ | bneck, 5x5 | 120 | 40 | ✓ | RE | 1 |
| $28^2 \times 40$ | bneck, 3x3 | 240 | 80 | - | HS | 2 |
| $14^2 \times 80$ | bneck, 3x3 | 200 | 80 | - | HS | 1 |
| $14^2 \times 80$ | bneck, 3x3 | 184 | 80 | - | HS | 1 |
| $14^2 \times 80$ | bneck, 3x3 | 184 | 80 | - | HS | 1 |
| $14^2 \times 80$ | bneck, 3x3 | 480 | 112 | ✓ | HS | 1 |
| $14^2 \times 112$ | bneck, 3x3 | 672 | 112 | ✓ | HS | 1 |
| $14^2 \times 112$ | bneck, 5x5 | 672 | 160 | ✓ | HS | 2 |
| $7^2 \times 160$ | bneck, 5x5 | 960 | 160 | ✓ | HS | 1 |
| $7^2 \times 160$ | bneck, 5x5 | 960 | 160 | ✓ | HS | 1 |
| $7^2 \times 160$ | conv2d, 1x1 | - | 960 | - | HS | 1 |
| $7^2 \times 960$ | pool, 7x7 | - | - | - | - | 1 |
| $1^2 \times 960$ | conv2d 1x1, NBN | - | 1280 | - | HS | 1 |
| $1^2 \times 1280$ | conv2d 1x1, NBN | - | k | - | - | 1 |

Table 1. Specification for MobileNetV3-Large. SE denotes whether there is a Squeeze-And-Excite in that block. NL denotes the type of nonlinearity used. Here, HS denotes h-swish and RE denotes ReLU. NBN denotes no batch normalization. *s* denotes stride.

Figure 2: MobileNetV3 (Large) Architecture (taken from [5])

*decoder with bicubic upsampling*. Both decoder architectures use 6 depth-wise convolution layers. The only difference is that the decoder with transposed convolutions uses point-wise convolutions for upsampling the reconstruction.

The only assumption made is that the feature maps can be stolen or intercepted when they are sent over the network. There are several attacks like MITM, replay attack, etc. that are suitable for this task, but this is not the focus of the problem discussed in this paper.

## 4.1 Workflow and Training Parameters

Here is the general workflow for generating reconstructed images according to the methodology discussed in this section:

- Preprocess input images by resizing and normalizing the data as per the model requirements.
- Use a pre-trained MobileNetV3 model as the encoder for the Autoencoder. This part of the model is frozen since it is part of the Trust Model.
- Initialize two decoder architectures: decoder with ConvTranspose and decoder with Upsampling+Conv
- Train the autoencoder according to the hyperparameters presented later in this section.
- Clip and unnormalize the output of the decoder to generate a valid image.

Here are the training parameters used to train both decoder architectures:
- Optimizer: Adam (default betas)
- Scheduler: StepLR (gamma = 0.93, step=3)

- Batch Size: 128
- Learning Rate: 2.5e-4
- Number of epochs: 25 epochs
- Loss Function: MSE Loss
- Additive Gaussian Noise during augmentation (variance = 0.02, mean = 0.02)
- Split: 10:1 training to validation split on 40k samples, single-fold

# 5 Quantitative Validation Method

While Mean-Squared Error is used as the loss function to optimize the model, it is not a reliable metric to determine the quality of reconstruction. To validate the quality of the image reconstruction, we evaluate the Structural Similarity Index Measure (SSIM) of the reconstructions. Due to errors in reconstruction, some of the high-frequency information may be lost, which can affect the SSIM value. Thus, we also account for a multi-scale approach for a fair evaluation (not discussed in [3] and [4]). The three Quantitative metrics for validating the performance of the model are:

1. Structural Similarity Index Measure (SSIM)

2. Multi-Scale Structural Similarity Index (MS-SSIM)

3. Peak Signal to Noise Ratio (PSNR)

In Equation 1, $\mu_x, \mu_y$ are the pixel sample means of $x$ and $y$. $\sigma_x^2$ and $\sigma_y^2$ are the variances of $x$ and $y$. $\sigma_{x,y}$ is the covariance of $x$ and $y$. $C_1$ and $C_2$ are constants for numerical stability.

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + C_1) + (2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{1}$$

In Equation 2, $M$ is the maximum scale considered, $c_j$ and $s_j$ are the contrast comparison and structure comparison at the $j^{th}$ scale. $l_M$ is the luminance comparison at scale $M$ and is only computed once. See [6] by Bovik for more details on the metric.

$$\text{MSSSIM}(x, y) = [l_M(x, y)]^\alpha \cdot \prod_{j=1}^{M} [c_j(x, y)]^{\beta_j} [s_j(x, y)]^{\gamma_j} \tag{2}$$

In Equation 3, $d$ is the bit-depth for an $N \times N$ size image. $x_i$ is the $i^{th}$ pixel of the original image and $y_i$ is the $i^{th}$ pixel of the reconstructed image.

$$\text{PSNR} = 10 \log_{10} \frac{(2^d - 1)^2 N^2}{\sum_{i=1}^{N \times N} (x_i - y_i)^2} \tag{3}$$

The maximum value for SSIM and MS-SSIM is 1, which means that the two input images are identical in terms of structural similarity. Values closer to 1 are indicative of high-quality reconstructions.

| Model Architecture | PSNR | SSIM | MS-SSIM |
|---|---|---|---|
| Baseline 1 | 58.65 dB | 0.6 | - |
| Baseline 2 | 62.3 dB | 0.7 | - |
| Decoder with ConvTranspose | 46 dB | 0.033 | 0.22 |
| Decoder with Upsampling+Conv | 48.6 dB | 0.039 | 0.24 |

Table 1: Reconstruction Quality of Different Decoder Architectures against Baselines

# 6 Evaluation

In Table 1, Baseline 1 refers to the work presented in [3] by Oh and Lee. Baseline 2 refers to the work presented in [4] by Benkraouda and Nahrstedt. Furthermore, the baseline methods do not use MS-SSIM, but I have used it to get better insight into the performance of the adversarial decoder.

We observe that the reconstruction quality of our models does not surpass the reconstruction quality of both baseline methods in terms of PSNR and SSIM. This is to be expected because this paper specifically focuses on attacking models that have complex architectures and skip connections. However, the reconstructed images are able to reveal enough information for it to be a privacy concern as seen in Figure 12. The SSIM scores for the proposed methods are low because they are limited to a single scale. The MS-SSIM measure relaxes this constraint, and as such, is more indicative of the similarity between the images.

Now, we can compare the performance of the two different decoder architectures. It is clear that the decoder with upsampling incorporated in the network outperforms the decoder using transposed convolutions. As seen in Figure 12, we can see that there are no checkerboard artifacts during reconstruction when using bicubic upsampling. This is because the kernel size and stride are not equal in the encoder. From an adversarial perspective, we do not have control over the model architecture. Thus, using the decoder with upsampling is a better choice.

We can visualize how the input gets reconstructed through each layer of the decoder (using the upsampled decoder since it performs better). Consider a sample image from the dataset in Figure 3.

Now, for the sample image in Figure 3, we can visualize the feature maps (intermediate representation). These feature maps are what the encoder (victim edge device) generates and will be intercepted over a network. A few of the feature maps are shown in Figure 4.

We have 7 layers before the final reconstructed output if we include the upsampling and convolution layers. Figures 5, 6, 7, 8, 9, 10, and 11 show the process of reconstruction through each layer. Note that as we get closer to the original layer, the effect of bicubic interpolation is more pronounced.

Figure 3: Sample from CelebA for in-depth analysis of the decoder

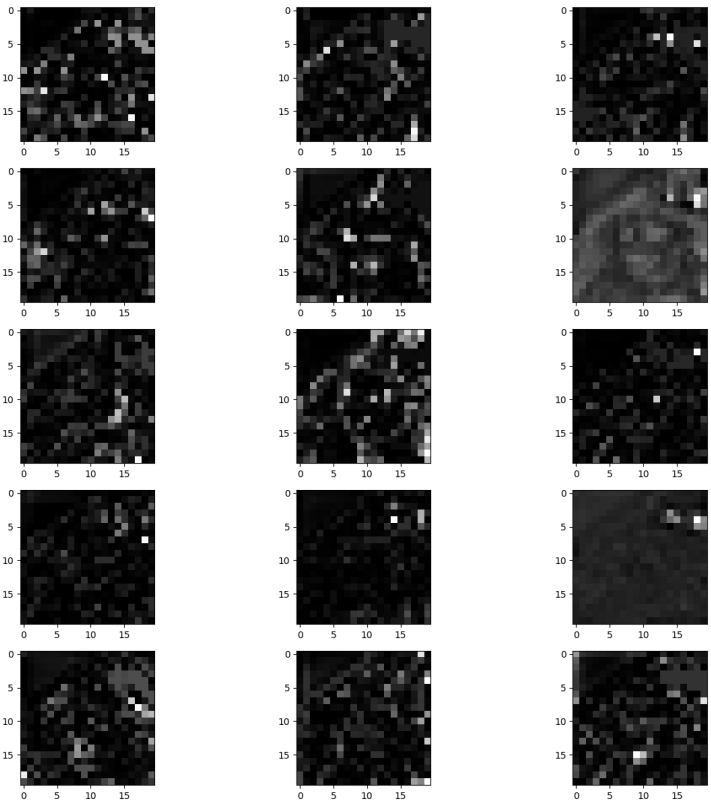


Figure 4: IR of Sample Image in Figure 3

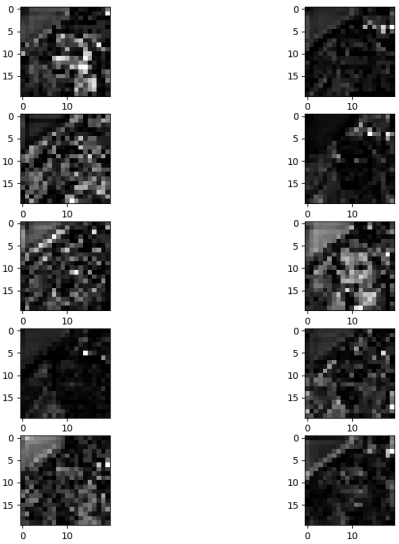


Figure 5: Layer 1 of Decoder

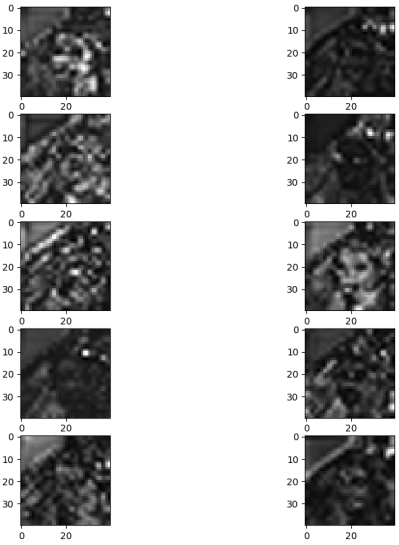


Figure 6: Layer 2 of Decoder

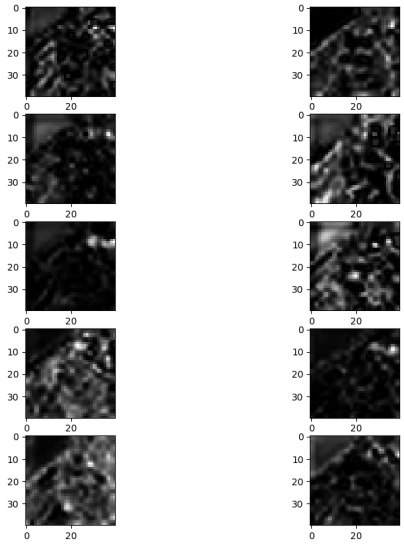


Figure 7: Layer 3 of Decoder

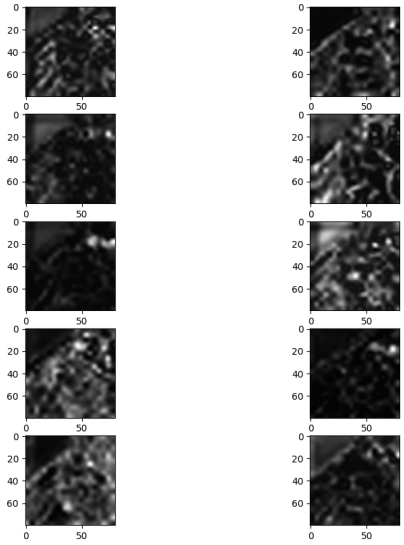


Figure 8: Layer 4 of Decoder

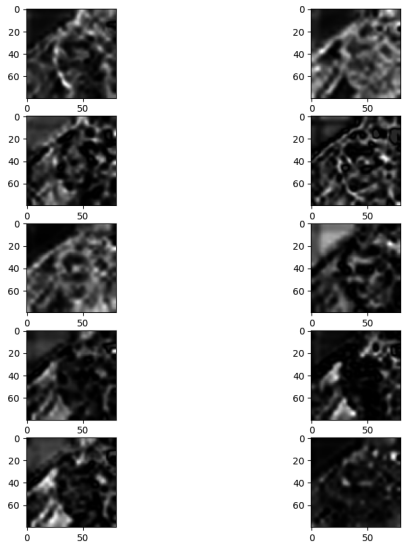


Figure 9: Layer 5 of Decoder

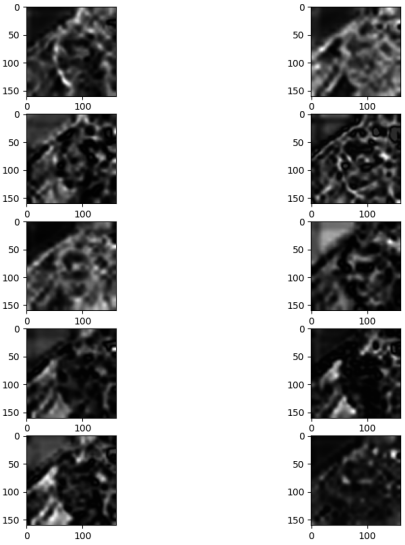
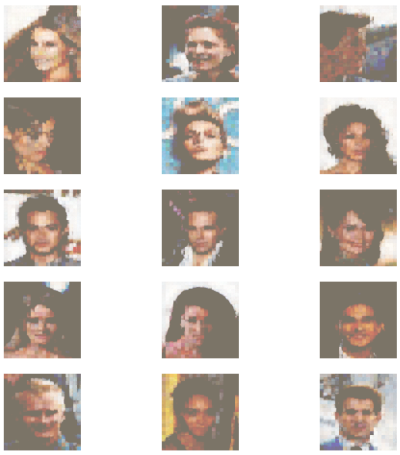


Figure 10: Layer 6 of Decoder

Figure 11: Layer 7 of Decoder

(a) Reconstructed Results using Decoder with Bicubic Upsampling



(b) Reconstructed Results using Decoder with Transposed Convolution

Figure 12: Results of Reconstruction using Adversarial Decoders

# 7 Discussion

As seen in the previous section, the proposed attack can recover sensitive information from intermediate representation of distributed deep learning architectures. To mitigate the effectiveness of the attack, sending a subset of feature maps at the cost of accuracy is a viable solution that ensures real-time performance. Note that I have not implemented a model that uses this modified architecture. In fact, one of the biggest disadvantages of this mitigation technique is that it cannot use standard pre-trained weights since the architecture is fundamentally different.

My future work aims to explore how privacy can be preserved in distributed deep learning models with real-time constraints and prevent attacks similar to the black-box attack proposed in this paper.

# References

[1] X. Wei, J. Zhu, S. Yuan, and H. Su, "Sparse adversarial perturbations for videos," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, pp. 8973–8980, Jul. 2019. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/4927 2

[2] X. Wei, S. Liang, N. Chen, and X. Cao, "Transferable adversarial attacks for image and video object detection," 2019. 2

[3] H. Oh and Y. Lee, "Exploring image reconstruction attack in deep learning computation offloading," in *The 3rd International Workshop on Deep Learning for Mobile Systems and Applications*, ser. EMDL '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 19–24. [Online]. Available: https://doi.org/10.1145/3325413.3329791 2, 3, 4, 6, 7

[4] H. Benkraouda and K. Nahrstedt, "Image reconstruction attacks on distributed machine learning models," in *Proceedings of the 2nd ACM International Workshop on Distributed Machine Learning*, ser. DistributedML '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 29–35. [Online]. Available: https://doi.org/10.1145/3488659.3493779 3, 6, 7

[5] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q. V. Le, and H. Adam, "Searching for mobilenetv3," 2019. 4, 5

[6] Z. Wang, E. Simoncelli, and A. Bovik, "Multiscale structural similarity for image quality assessment," vol. 2, 12 2003, pp. 1398 – 1402 Vol.2. 6