Image quality improvement with autoencoders

02456 Deep Learning final project

Karol Krzak
s203005@student.dtu.dk

Jerzy Nawrocki
s202618@student.dtu.dk

Maciej Tatarski
s202609@student.dtu.dk

ABSTRACT

Images are used in many different areas, and their quality plays a crucial role in their usefulness. We present a study on several ways of approaching the improvement of image resolution, primarily focused around convolutional autoencoders. We provide information about data sets CIFAR10 and Cats&Dogs we have been working with and necessary introduced modifications. We discuss changes in the use of the most widely used loss criterion and explain why and what we have used instead. Further, we introduce used neural networks architectures: convolutional autoencoder and decoder as well as a residual neural network, the latter one obtaining the best results overall. In the end, we compare results and discuss future work.

1 INTRODUCTION

The quality of images plays a crucial role in both human and automated decision making processes from security, military operations, mobile vehicles path planning to medical diagnostics and quality control. Higher quality images allow for better problem understatement, therefore more self-aware/well-informed choice of actions. Unfortunately, often the reason behind lower picture quality is the price of the state-of-the-art equipment which would allow obtaining the best possible results. Thus, in some cases, cheaper sensors are used in capturing the desired data. In addition, conditions in which images are taken (low light, shade, camera movement) leave much to ask for, making the situation worse. Hence, the above mentioned application areas have much to gain from more robust and efficient image quality improvement algorithms.

In recent years, deep learning approaches have gained a lot of followers due to their superior performance in comparison to other image processing methods. Neural networks, ranging from Convolutional (CNN) to recent Generative Adversarial Nets found the most success in feature extraction, segmentation, denoising, and quality improvement. That is why our focus was put on these strategies to explore them and find out their potential.

In this paper, we will work with different deep learning architectures with a primary focus on autoencoders to achieve better image quality. First, we will introduce a real-life problem that is the motivation behind these experiments and everything related to the data sets we have been using including applying random salt & pepper noise. In the further sections, we will deliberate on models we have used and for a specific task in the order, they were deployed. Finally, we will present conclusions from our work and the possible direction of future work in this field.

2 PROBLEM STATEMENT

Omhu is a company which is developing digital solutions in dermatology and is revolutionising patient experience. Their product aims to enable anyone in the world with skin conditions to have access to a dermatologist. Combining technology with professional expertise they provide new ways of delivering healthcare to those who need it most, anywhere.

Their clients - patients can now photograph their skin conditions and upload them to the Omhu app. There, behind the scenes machine learning algorithms together with medical professionals assess received data in order to properly determine patients' condition. Unfortunately, those pictures offer suffer from previously mentioned issues, resulting in their low quality that prevents accurate diagnosis.

Our goal is to develop a type of neural network known as autoencoder that would solve this problem by removing each photograph issue and enhancing its quality so that it would be feasible to determine patients' condition. By manually applying transformations which imitate common issues with received patient's photographs to the data set of good quality images, we will train our network to reconstruct its former properties. That would enable specialists and other networks used during diagnosis to work efficiently despite significant variation in received data quality.

3 DATA PREPARATION

This section fully explains how both datasets were prepared for training, validating, and testing processes. The first subsection is devoted to the CIFAR10 dataset used for denoising, next Cats and Dogs is used for the image quality.

3.1 CIFAR10

For the first part of our project it was decided to introduce a noise to the images in order to build the model for denoising photos.

As initial dataset was chosen to be CIFAR10 dataset. It consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images[4]. Original photos haven't had any noise so artificial one was introduced. As for types of noise, 4 main types were used:

- **Salt and pepper** - random pixels were chosen with a given proportion of noise to be either black or white(255,255,255 or 0,0,0 in rgb).
- **Random colorful noise** - random pixels were chosen to be colored with a random color.
- **Darkened image** - in all images values of the pixeles were lowered for a certain percentage to achieve the effect of darker photos.
- **Brightened image** - in all images values of the pixeles were multiplied for a certain percentage to achieve the effect of darker photos.

Proportion of noise was set between 30-60 % in all models. Example of noises can be seen on figure 1.

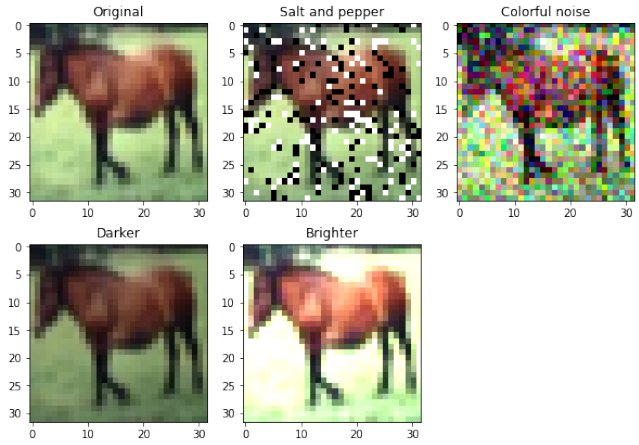


Figure 1: Example of noises.

3.2 Cats and Dogs

This dataset has been used for the resolution enhancement task, it contains 25000 labeled images with different sizes. Labels of images for this problem are not needed, thus not used.

To decide what input and output size to choose we first calculated the average size of all images, and based on this information we decided that the optimal output image size will be 360x360 and input size will correspond to model definition (either 180x180 or 90x90). Moreover, to fit the given size 360x360 we decided that if the image is bigger than this size then the center crop is performed on an image. On the other hand, if an image is smaller, then rescale is used with bicubuc interpolation.

3.3 Losses

During the development of the resolution enhancement model, it was clear that the most popular criterion (MSE) was producing blurry images, even with the fact that the error was fairly low. To fight this, a few loss functions were introduced and tested, to see if this will have any impact on the model itself. Below there are listed losses which were used in this project.

- **MSE** - (Mean Square Error) most used criterion in any type of application. With this one models tended to reproduce images with blur. $MSE = \frac{1}{n}\sum(X_i - Y_i)^2$. [2].
- **PSNR** - (Peak Signal Noise Ratio) where MAX indicates maximum value of pixel, in this case it can be 255 or 1 if data is normalized. This loss appeard to perform similar to loss mentioned previously, due to the fact that main error is computed in the same way. $PSNR = 10 \log \frac{MAX^2}{MSE}$. [2].
- **SSIM** - (Structural Similarity Index Measure Loss) - this loss is kernel based, where instead computing loss between each individual pixel we compute difference between two kernels with given size. This measurement is taking in to account luminescence, contrast and structural difference. $SSIM(x, y) = \frac{(2\mu_x\mu_y+c_1)(2\sigma_{xy}+c_2)}{(\mu_x^2+\mu_y^2+c_1)(\sigma_x^2+\sigma_y^2+c_2)}$. [7].

4 MODELS

Several models were tested but only three of them were chosen for the final results. All of the models and utilities are stored in a GitHub repository[6]. All parameters of the models, as well as implementation and scripts for data processing and visualization, are included in the repository README.md file and can be accessed through the GitHub repository. A complete diagram that shows information flow for training and visualizing the model can be seen in appendix A.

4.1 Convolutional autoencoder

Two models of autoencoders were created to tackle noisy images problem - convolutional and variational autoencoders. The former was giving better results as well as was easier to implement so it was decided to move forward with this model.

A convolutional autoencoder is a neural network that is trained to reproduce its input image in the output layer. An image is passed through an encoder, that produces a low-dimensional representation of the image. The decoder takes this compressed image and reconstructs the original image.

The final model for the encoder consisted of 2d convolution followed by 2d maxpooling and two more convolutions. The decoder was designed as a mirrored encoder, but instead of convolutions and maxpooling it used transposed convolutions and upsampling. A diagram of the final model can be seen in figure 2. All convolutions used kernel size equal to 4 and stride equals 2 so the size of the outcome is half of the one before convolution.

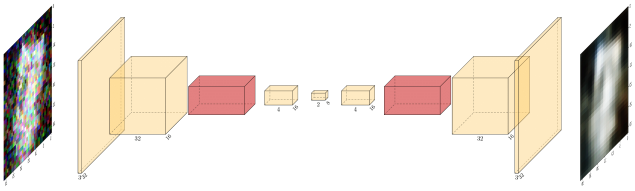
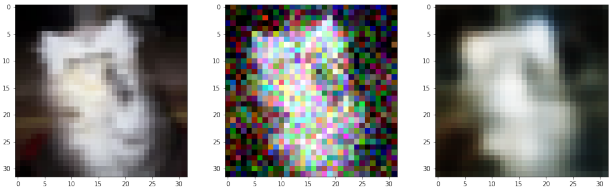


Figure 2: Auto-encoder model diagram

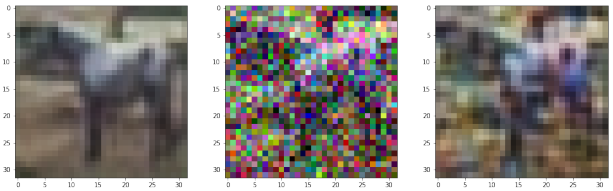
In the initial model, there was a fully connected dense layer between encoder and decoder but it was quickly found that it was producing a worse outcome than the one without it. The final size of the bottleneck - the smallest representation of the images was set to 8x8 with two channels. This size was chosen as it was significantly smaller than the original image so it prevents over-fitting as well as it still contains enough data to be able to reconstruct the image.

Two approaches were considered for model training - on original not noised data (model A) and noised data (model B). Results from those two approaches are shown on figure 3 and figure 4. The model trained on noised data was trained on images with the addition of colorful noise as input but the output was compared to pictures without a noise.



(a) (b) (c)

Figure 3: Results from autoencoder trained on original images(model A). (a) Original (b) Noised (c) Reconstruction



(a) (b) (c)

Figure 4: Results from autoencoder trained on noised images(model B). (a) Original (b) Noised (c) Reconstruction

Model trained on the data without the noise (model A), overall seemed to be perform better. Model B tended to produce colorful spots in the image which did not resemble real colors on the picture. Additionally, it was discovered that another benefit of training the model on clear, not noisy photographs was that the model A was also performing well in the reconstruction of images that did in fact have implemented some sort of noise. That was not the case with model B.

4.2 Deconvolutional neural network

Deconvolutional neural network, which is transposed convolution operator over an image, was used for image resolution enhancement after successfully tackling the problem of image denoising. In this problem, we assumed that our models do not need an encoder, but only a decoder and that bottleneck will be our input. Model shown in the Figures 5 and 6 was introduced to check if even a small and simple decoder for resolution enhancement will work. As it turns out, it could successfully reconstruct an image, but the quality of it is similar or even worse than nearest-neighbor interpolation, therefore there is still ground for improvement.

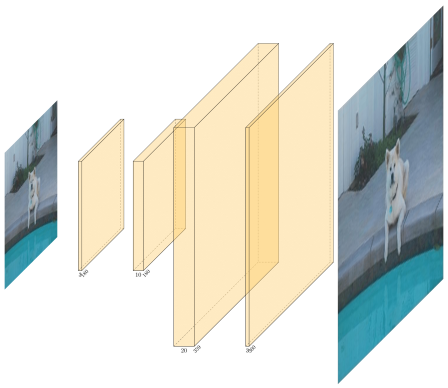


Figure 5: Deconvolutional model diagram.

In Figure 6, it is shown that the input of the model is an image with size 180x180 and the desired output is with size 360x360. In order to increase the size of the photo from 180x180 to 359x359 (layer 2) we used kernel size 3 and stride 2, and in the first layer to keep the size the same we used kernel size 3 and stride 1. In the last layer, in order to match the size of output 360x360 we used kernel size 4 and stride 1. The number of channels used in this model is respectively 10, 20 and 3, where the last one has to have 3 channels because we want an RGB image for the output.

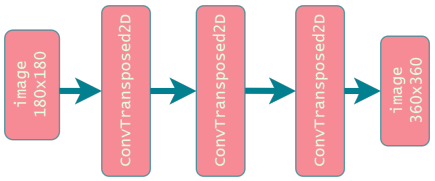


Figure 6: Deconvolutional model diagram.

Example results from our first model can be seen below, where there are 3 images side by side to compare the results between ground truth, model output, and also to see what is the image quality before feeding it to the trained model. It seems hard to distinguish any differences, but in order to do that, we need to look closer at the edges of the dog's ears or dog's back. In those spots we can notice the difference.

Figure 7: Example results from Deconvolutional Neural Network. (from left to right 180x180 model input, 360x360 model output, 360x360 ground truth)

4.3 Residual neural network

A very interesting implementation of convolutional layers is their incorporation in residual neural network architecture, a relatively recent, innovative approach proposed in this research paper from 2015 [1].

While working with neural networks related to computer visions, people tend to implement more and more layers. That is done to achieve better results in solving highly complex problems, as each layer can be trained to extract different features and in the end obtain an accurate representation of the input data. With an increasing number of layers, one will arrive at a point when several problems can be encountered - those being vanishing gradient and degradation. The neural network's weights are updated with the value proportional to the partial derivative of the error function relative to the current weight in every interaction of the training. With an increasing number of layers, the gradient might become incredibly small, which will result in constant weight values or in worst cases complete prevention of training [3]. As the depth increases, the network's accuracy saturates - as at some point it will learn all our data, and degrades rapidly. In result. we obtain worse training error compared to the shallower net.

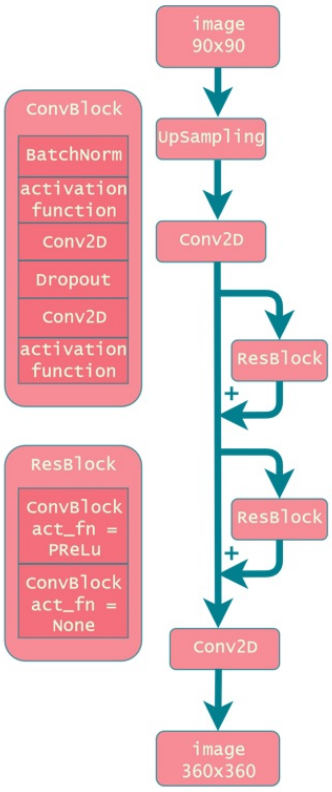


Figure 8: Residual neural network with convolutional blocks.

To counter those issues residual neural networks implement identity shortcut connections to jump over some layers. Typically ResNet models utilize double or triple layer skips. The shortcut connections perform identity mapping and are adding the output of the last non-skipped-layer to the outputs of stacked layers. That way no additional parameters are created and the computational complexity remains the same.

In our model we decided to use most common approach to residual networks with double skipped layers, proceeded by batch normalization and ReLu activation. In addition, we have implemented a dropout layer between two skipped convolutional layers. Furthermore, we decided to combine the mentioned above elements into what we have called "Convolution blocks" to make implementation more convenient. After that they themselves were stack into bigger blocks called "Residual" The skip connections is used on the "ResBlocks", which our model has two in total. Apart from that, the model has two free standing 2D convolutional layers and an upsampling one at the very beginning. The image fed to the network had originally size of 90x90 pixels, with the output being 360x360.

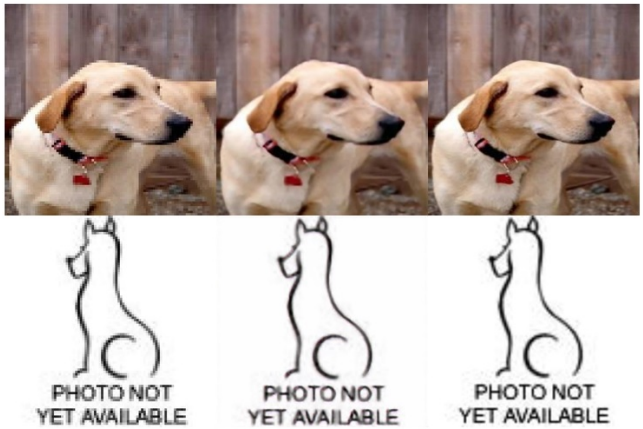


Figure 9: Example results from ResNet. Left: low resolution image. Middle: reconstruction. Right: ground truth image.

The results can be seen above. It can be observed immediately that our model did in fact enhance the quality of the low resolution version of the dog photograph. in comparison to the left image, reconstruction has no visible pixels and the borders between contrasting colors are smooth. Although the reconstruction isn't as sharp as the the ground truth it is still a significant improvement in coverall quality.

5 RESULTS

With the results from the image denoising, we can conclude that the convolutional autoencoders seemed to perform significantly better over VAE and AE with a fully connected layer between encoder and decoder. Moreover, training this model with no noisy data at all could far better reconstruct the original photo rather than training it with noisy data, where the model was producing photos with bad colors and shapes. Since we were not pursuing this path of the project anymore we assumed that current results are satisfactory, nevertheless, there can be more improvements done by using more advanced architectures like U-Net.

	MSE	PSNR	SSIM
NNI	27.25	0.8717	0.0024
Deconv	26.19	0.8863	0.0030
ResNet	28.26	0.8959	0.0020

Table 1: Deconvolutional Neural Network (Deconv) and Residual Neural Network (ResNet) performance compared with with Nearest Neighbour Interpolation

In the image resolution enhancement part,in order to visualize the differences in performance between our two models (Deconvolutional and ResNet) and already known, popular solution (nearest-neighbor interpolation - NNI) in image analysis a table was created. It contains computed errors using 3 different types of losses (MSE, PSNR, SSIM) which were explained in section 3.3. From it, we can see that the Residual Neural Network (ResNet) is performing better than the other two with every loss function. We can also see that Deconvolutional Neural Network is performing better with MSE and PSNR but with SSIM it is worse. That means that for human eye DecNN reconstruction is worse or very similar to that of the NNI, despite seemingly better results with MSE. That is indicated by the SSIM results.

6 CONCLUSIONS

In this project, two models for image denoising and two models for image resolution enhancement were introduced. All goals defined in the project scope were achieved.

6.1 Image denoising

Denoised images produced by convolutional autoencoder model were significantly better compared to the variational autoencoder and the original noised ones. It was discovered that additional dense layer in the middle of autoencoder architecture makes the image quality visibly worse. Best performance was achieved using bottleneck of the size 8x8x2, with this size model was not prone to overfitting and underfitting.

6.2 Image quality improvement

Image resolution enhancement turns out to be a complex and time-consuming task, because not always low MSE loss indicated a good reconstruction, therefore we were researching various approaches that other uses in similar studies. We ended up trying different loss functions as well as different more advanced neural network architectures. Final model was decided to be Residual Neural Network as the identity connection between blocks had a positive impact on the reconstruction of an image. Additionally, we were quite surprised that even a Deconvolutional Neural Network can match the performance of Nearest Neighbour Interpolation.

To speed up the process of developing and training various models we created a pipeline that utilizes HPC from DTU and automates repetitive tasks. This, in the beginning took some time, nevertheless we were able to train, save and evaluate far more models, than if we would do this manually. A diagram that shows this process can be found in the appendix A.

6.3 Future Work

Timeline of the project and current state can be seen on figure 10. As for now project is in stage when some additional tweaking for residual neural network model would be needed to achieve even better results.

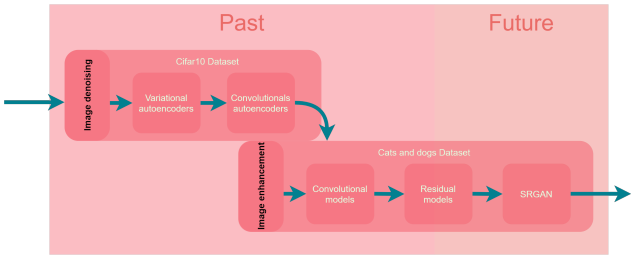


Figure 10: Plan for future.

For the next steps, it was discussed that the most state of the art approach would be to implement generative model for resolution enhancement such as SRGAN which is explained in the section below.

6.4 SRGAN

SRGAN is a generative model for super resolution of the images. It is solving the central problem of other super-resolution models which is texture details recovery. It is using generative adversarial network (GAN) implementing generator and discriminator as well as custom loss function which consists of an adversarial loss and a content loss.[5]

REFERENCES

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep Residual Learning for Image Recognition. (2015). https://openaccess.thecvf.com/content_cvpr_2016/html/He_Deep_Residual_Learning_CVPR_2016_paper.html

[2] Alain Horé and Djemel Ziou. 2010. Image quality metrics: PSNR vs. SSIM. 2366–2369. https://doi.org/10.1109/ICPR.2010.579

[3] Fakultit Informatik, Y. Bengio, Paolo Frasconi, and Jfirgen Schmidhuber. 2003. Gradient Flow in Recurrent Nets: the Difficulty of Learning Long-Term Dependencies. *A Field Guide to Dynamical Recurrent Neural Networks* (03 2003).

[4] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. [n.d.]. CIFAR-10 (Canadian Institute for Advanced Research). ([n. d.]). http://www.cs.toronto.edu/~kriz/cifar.html

[5] Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. 2017. Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network. arXiv:1609.04802 [cs.CV]

[6] own. 2021. DL-IQIWAn. https://github.com/mactat/DL-IQIWA.

[7] Zhou Wang, Eero P. Simoncelli, and Alan C. Bovik. 2003. Multi-Scale Structural Similarity for Image Quality Assessment. In *in Proc. IEEE Asilomar Conf. on Signals, Systems, and Computers, (Asilomar.* 1398–1402.

A MLOPS

Created models were large, so it was decided to train them on DTU HPC. Below on the figure 11, process of training the models, storing artifacts and visualizing output is shown.

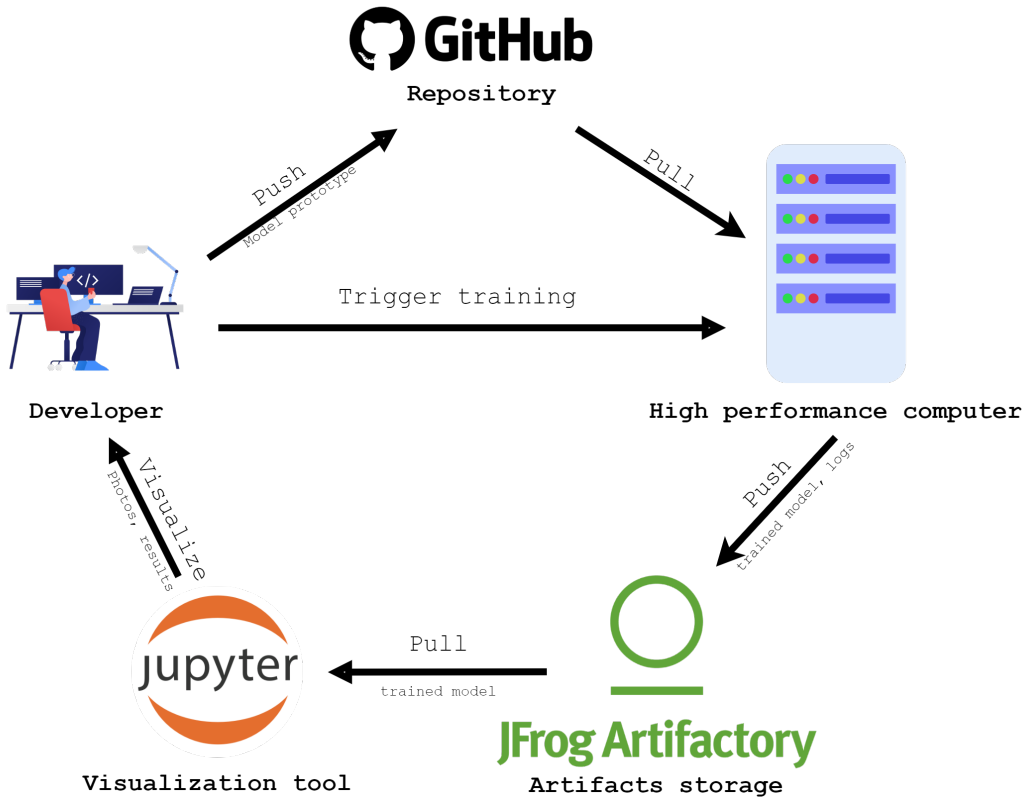


Figure 11: Training of the model.

B HIGHER RESOLUTION PHOTOS

Figure 12: Results of convolutional autoencoder.

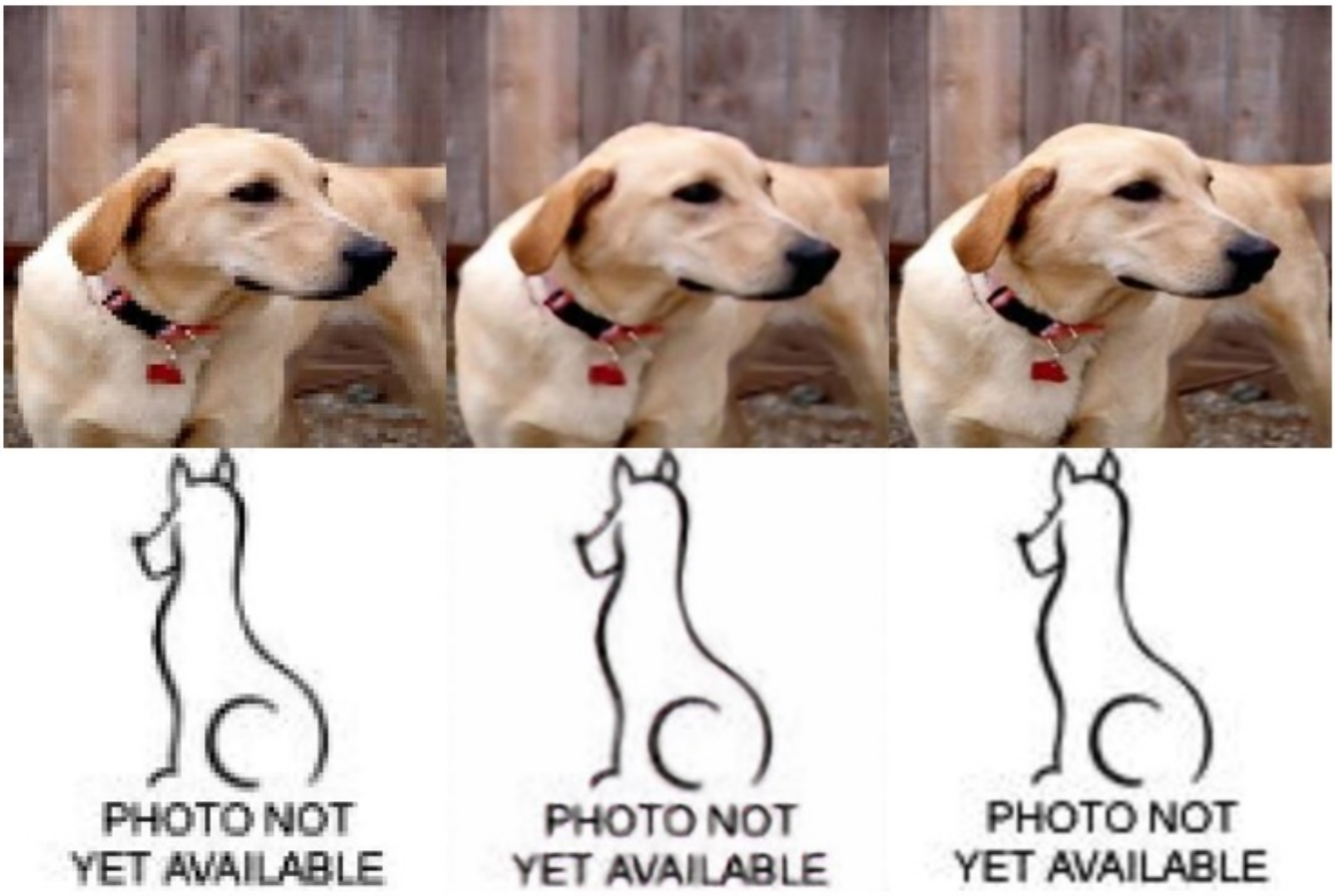


Figure 13: Results of residual neural network.