Denoising and Deblurring Images using Dual Autoencoders

CS-512 Computer Vision Project Report

Yash Khandelwal

Raghukarn Sharma

04.17.2023

Spring 2023

Task Delegation and Member Responsibility

Yash Khandelwal

- Implemented the data preprocessing steps including noisy images generation
- Built and experimented model architectures in Keras for multiple models
- Performing model training and hyperparameter training for those models
- Implemented the model evaluation and testing code
- Presentation and Report

Raghukarn Sharma

- Exploration and analysis of multiple datasets
- Defined data loading pipeline
- Built and experimented model architectures in Keras for multiple models
- Performing model training and hyperparameter training for those models
- Presentation and Report

Problem Statement

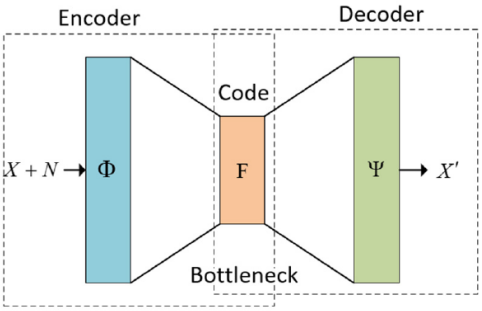
Image denoising is a process of removing noise or unwanted elements from a digital image. With the development of digital cameras, new types of noises that can affect an image have appeared, making image denoising a significant area of research. Image denoising is used in various fields such as image restoration, visual tracking, image registration, and image segmentation. Artificial intelligence and machine learning have found origins in image denoising because images are one of the main sources of information that can arise from different origins. There are various algorithms and approaches to achieve image denoising, such as spatial domain filtering, total variation, non-local means, data-adaptive transform, non-data-adaptive transform, and supervised learning. Image denoising can be thought of as a supervised learning problem where each noisy image represents an input, and the intended result is the original clean image. The fact that the processing of the inputs for any computer vision task such as object detection, image segmentation, object classification etc. requires high quality images for better and more accurate results but the images being highly sensitive to noise generated from internal and external camera parameters doesn't make that easy for us. The only hope we have is to clean such images before actually passing them for any computer vision algorithm to work on in order to get the expected results. Motivated by this idea, we are going to explore and experiment with multiple autoencoder architectures for various types of noise such as Gaussian noise, Poisson noise, Speckle noise, Impulse noise and further evaluate them on the basis of structural similarity and signal to noise ratio. The models being evaluated are the ordinary autoencoders against the dual autoencoders with and without separable convolutions. Moreover, we also try to introduce skip connections in the dual autoencoder architecture for better performance as they do in some of the well known architectures for object detection and image segmentation.

Proposed Solution

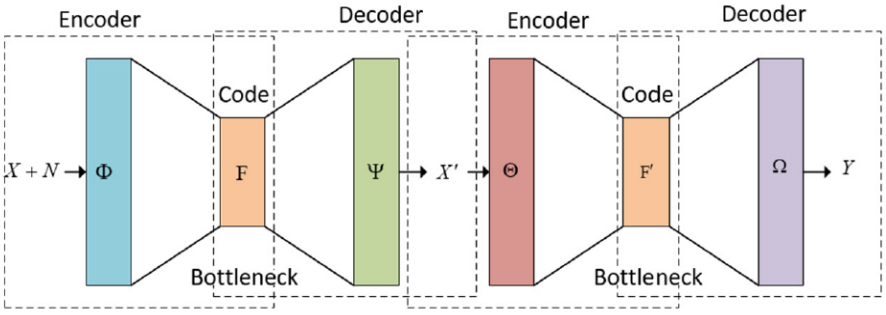
Over the years many neural network architectures have been proposed that are capable of performing image denoising / deblurring. The most common neural networks for such a task are autoencoders. Autoencoders are a kind of neural network that are utilized to develop efficient coding for unlabeled input i.e these models work on unlabeled data which brings them under the category of unsupervised learning as no specific labels are needed to train the network. The idea is to drive the input through a bottleneck which is a latent representation of the input data which is represented in a compressed manner. The fact that reconstruction of the input image from the bottleneck is not accurate as we have already lost a lot of spatial information amongst the image pixels our hope is that the model will eventually learn to deliberately lose the noisy pixels and instead fills them with the relevant pixel values during reconstruction that are directly related to the structure in the input data such as the correlation between input characteristics. This structure may be learned and used to our advantage when driving the input

through the bottleneck in the network.

Generally speaking, the autoencoders are performing feature extraction and at the same time leveraging the pyramid feature exploration paradigm wherein everytime we extract features we drop down the spatial dimensions of the input as we increase the number of channels to make up for the loss of information. By doing so we eventually reach a bottleneck which is nothing but a latent representation of the input data combining all the most important features into a much smaller spatial map. From this point we try to reconstruct the original image as closely as possible minimizing the loss with the original image itself, and the model tries to learn the class of each pixel on its own unlike any image segmentation task where we have a specific labeling for each image pixel. This architecture works quite well in most of the cases but the quality of reconstruction of image and the structural similarity between the clean image and the reconstructed image is not very high which leads to hinder the performance of the tasks that follow. To deal with this problem, we explore more advanced autoencoder architectures that deploy double layers of autoencoders in order to remove the noise but at the same time improve the image reconstruction quality. The image below shows a typical architecture of an autoencoder where the input image goes through an encoder and generates a bottleneck which then goes through a decoder and reconstructs the image of the size equivalent to the input image.



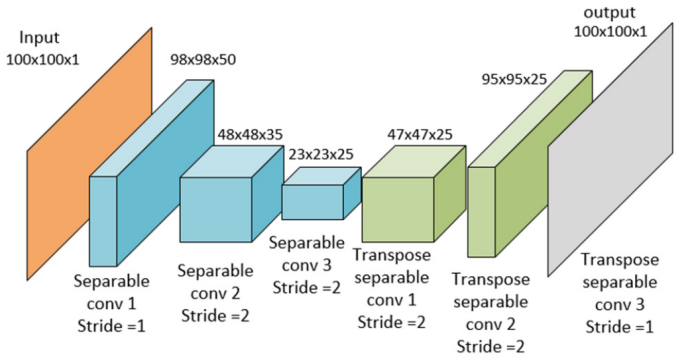
The image below is the dual autoencoder architecture proposed in the paper [1] wherein the first half is similar to the above shown ordinary autoencoder architecture but the second half takes the input from the first autoencoder and again drives it through the another bottleneck to reconstruct the image.



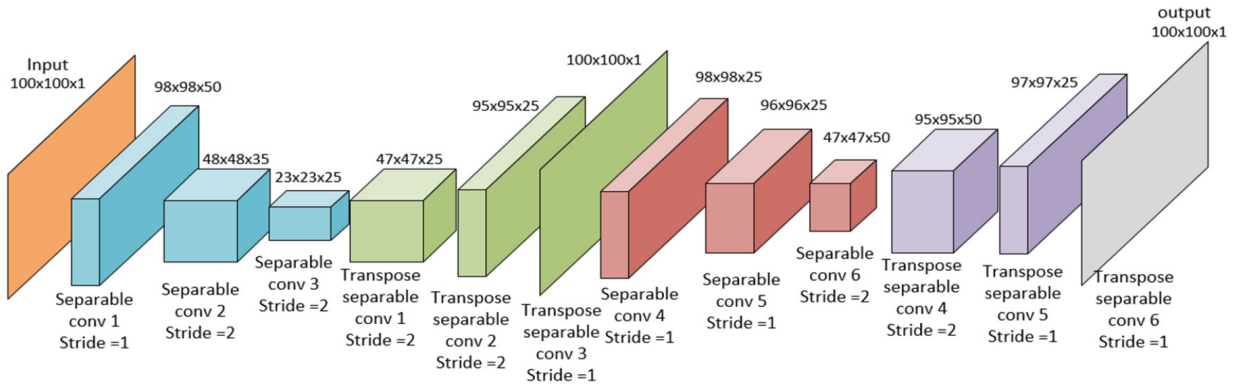
The intuition behind dual autoencoder is that the first autoencoder helps with the noise reduction which it

already does while the second autoencoder helps with image reconstruction quality i.e. it enhances the quality of the image such that we get a higher structural similarity with the original clean image.

With the introduction of the dual autoencoders we also introduce a lot of trainable parameters which inturn increase the training as well as the inference time of the model. To deal with this problem we replace all the convolutions with depthwise separable layers similar to the mobilenet paper [3] which reduces the number of floating point operations but an enormous amount leading to faster training and inference time but the trade off is they are a bit less accurate. Our analysis has accurately evaluated the performance comparison between the model with ordinary convolutions and the ones with separable convolutions. The exact architectures as defined in [1] can be seen below. The first is the ordinary autoencoder architecture that is implemented in this project which takes an input of size 100x100x1 and downscales it to a bottleneck of size 23x23x25 from where it upscales and reconstruct the image to get an output of 100x100x1.



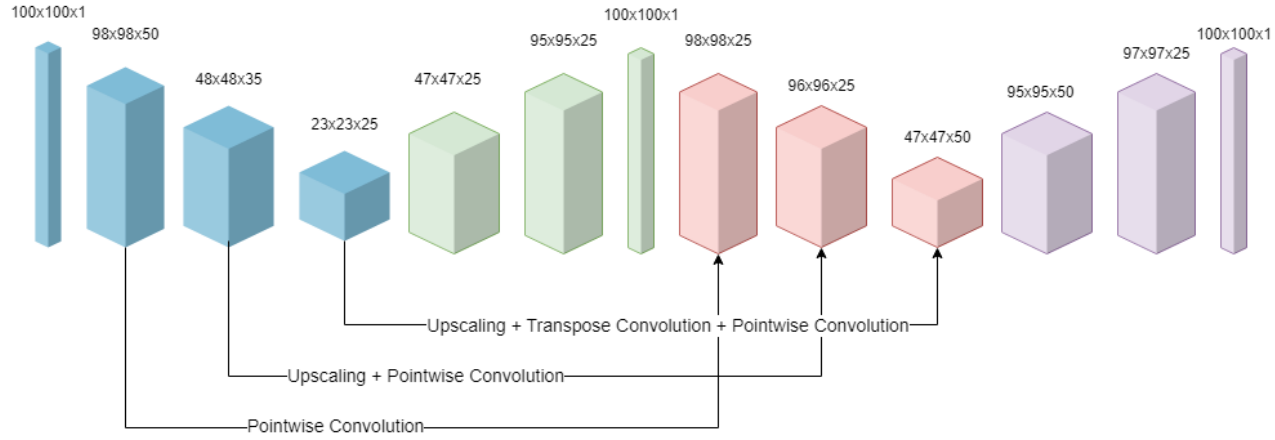
The below figure shows the architecture of the dual autoencoder which takes a 100x100x1 image as an input downscales it to 23x23x25 then upscales it to 100x100x1 which is exactly what happens in the ordinary encoder. But after that it again downscales it to 47x47x50 and finally we get an output of 100x100x1.



Both the above architectures have been implemented and evaluated with ordinary as well as separable

convolutions.

Finally, we also propose a variant of this dual autoencoder architecture which deploys skip connections in the network going from the first encoder into the second encoder. The data from the first encoder is upscaled / downscaled or increased / decreased in dimensions in order to match the dimensions of the respective layer in the second encoder. To keep the number of floating point operations under control we don't concatenate the data coming from the skip connections but rather we add both the layers just like the ResNet architecture [2]. This allows us to not increase the number of operations but at the same time it increases the amount of information we have to give us a better reconstructed image. This architecture is shown in the image below.



Implementation Details

Dataset Used : IMDB-WIKI Dataset [4]

Dataset analysis & Description : The IMDb Wiki Faces dataset is a large-scale dataset that contains images of faces along with associated metadata such as age, gender, and identity. The dataset contains over 500,000 images of faces, making it one of the largest publicly available face datasets. Each image is labeled with the person's name and an associated ID, which allows for the grouping of images belonging to the same individual. For our project we are only interested in the face images and not the metadata. We also explored the other 3 datasets mentioned in the paper i.e. FER-2013 [5], KDEF [6], and AffectNet [7] but because the IMDB dataset is easily available and it contains more images than we need to train our model we only use this dataset.

Dataset Preparation : The first step in data preparation is to choose the subset of the dataset that we selected and that is done randomly so as to make sure we get enough variations in our dataset and it includes all the types of face images and not just of some particular type. Once we get the images ready

our next step is to create noisy images to train the model. For the purpose of this project we add the following types of noise in our data :

- Gaussian Noise : This is a sort of signal noise with a probability density function equivalent to that of the normal distribution i.e. the potential values of the noise are dispersed according to the Gaussian distribution depending on a given mean and variance values. In our case the mean was taken as 0 while the variance was 10. The sample image can be seen below.

- Poisson Noise : The quantized character of light and the independence of photon detections make Poisson noise a primary kind of uncertainty related to light measurement. Except in low light settings, its predicted magnitude is signal-dependent, and it is the most common cause of picture noise. A sample image can be seen below.

- Speckle Noise : Speckle noise is a type of noise that appears as random variations in the intensity of an image, caused by the interference of scattered waves. It is often seen in images obtained from ultrasound or synthetic aperture radar systems. This again depends on a specified mean and variance which in our case were 0 and 0.02 respectively.

- Impulse Noise : Salt and pepper noise is another name for impulse noise where sharp and rapid disruptions in the visual signal might create this noise. It manifests as a scattering of white and black pixels. In our case the value was taken around 30% of the image pixels.

For all the above noise types the amount of noise was randomly generated between a predefined range. Moreover, for every image in our dataset 4 images were generated, so in total *we made a dataset of 200,000 pairs of noisy and clean images from the 50,000 images we selected*. All these images were arranged in different directories, so in total we had 5 directories one for the clean images while rest for each noise type. All the noise types were added to the images using *scikit-learn* and its *utils* module.

Dataset Preparation and Augmentation code file : data_preparation.ipynb

Implemented Model Architectures : The following models were implemented in Keras from scratch and the hyperparameters as well as the dataset used to train them was kept the same. Moreover, each of these models had 2 versions i.e. one with ordinary convolution and another with separable convolution. The models are as follows :

- Single / Ordinary Autoencoder : Same as mentioned in [1]
- Dual Autoencoder : Same as mentioned in [1]
- Dual Autoencoder with skip connections (ours)

Training Configuration and Details :

The number of *Epochs* is set to 40 for all the models whereas the *Input Image Size* and the *Batch Size* are set to (100,100, 1) and 128 respectively. Moreover, the *Learning Rate* was taken as 0.001 with *Adaptive moment estimation (Adam)* as the Optimizer. Furthermore, *Binary Cross Entropy* was used as the Loss Function to train the models with *Sigmoid* as the Activation function in the last layer as we are typing to classify labels for each pixel and take them as close as possible to the clean image. For the intermediate layers *ReLU* was used. Finally, the data split for training was 144003, testing was 40001 and validation was 16000 images. All the training related code can be found inside *cv_final_project.ipynb* file.

A data generator was created for loading and training the model. The generator loads a pair of noisy images and its corresponding clean image as grayscale images, reshapes them (100, 100, 1) and normalizes both of them by scaling them by 255 and also changing their datatype to *float32*.

Evaluation Metrics : The performance evaluation of any image denoising task is not as straightforward as it is for other image related tasks like classification as this is a self-supervised learning problem. So we have to use more sophisticated metrics such as *SSIM* and *PSNR*. SSIM is the structural similarity index

which is used to compare the similarity between the clean image and the restored image and it is calculated by taking into account the luminance, contrast and structure of the images whereas the PSNR is the peak signal to noise ratio which is used to estimate the quality of image restoration as it is calculated by measuring the ratio of the maximum possible power of the original signal to the power of the signal that has been distorted by the compression or distortion. For both these values the higher the score the better the model is as high value of SSIM is high similarity between the images whereas high value of PSNR means less noise. The range of SSIM is from -1 to 1 where 1 means most similar, 0 means not similar and -1 means totally different whereas theoretically PSNR varies between 0 and infinity.

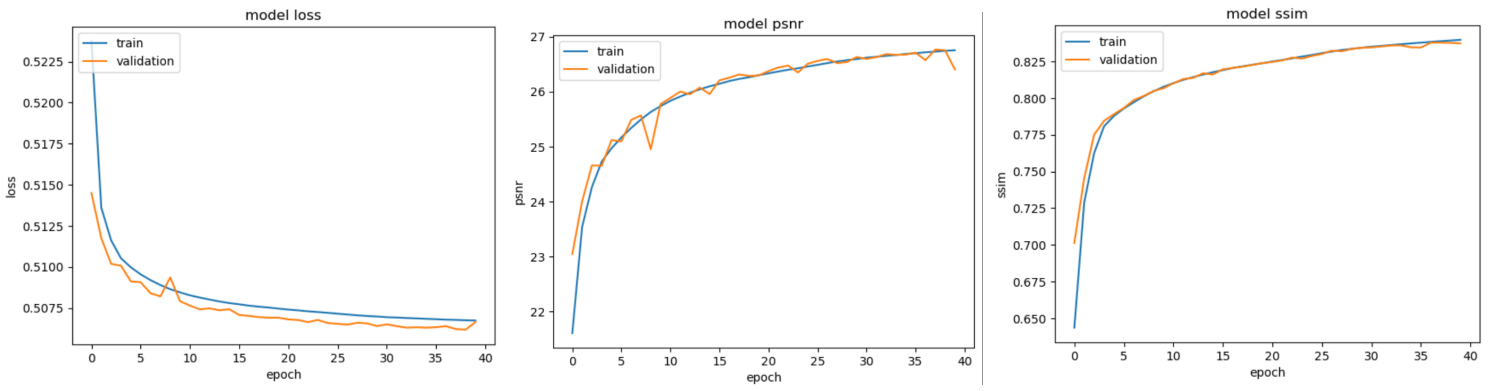
Testing Instructions : Steps to be able to test all the saved models on the test images is as follows :

- Set up the environment as mentioned in the *requirements.txt* file (!pip install -r requirements.txt)
- Download dataset zip and the trained models from the provided links in *data.txt* and *models.txt*
- All code for testing and evaluating the models is inside *testing.ipynb* file
- Run the top 4 code blocks inside this ipynb file to initialize all the modules that are needed
- Run the *test_models()* function for testing on the images and run the *evaluate_all_models()* and *evaluate_all_models_by_noise_type()* functions for evaluating on the test set. All the instructions for running the functions with different models and input images are given in comments in code

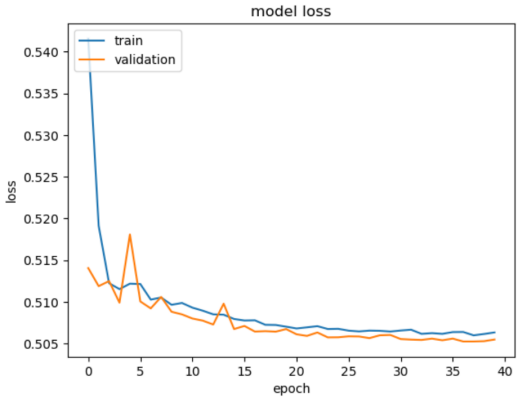
Results and Discussion

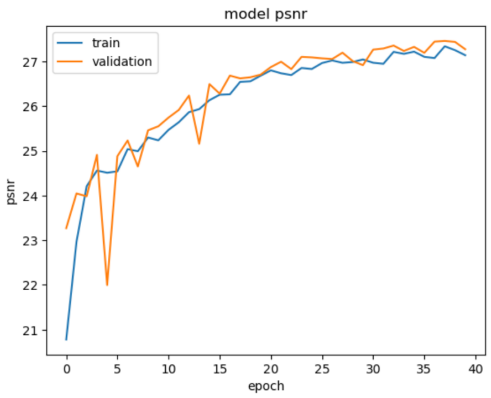
Dataset Used : As already discussed several models with different architectures were trained. Following are the loss, PSNR and SSIM plots of them:

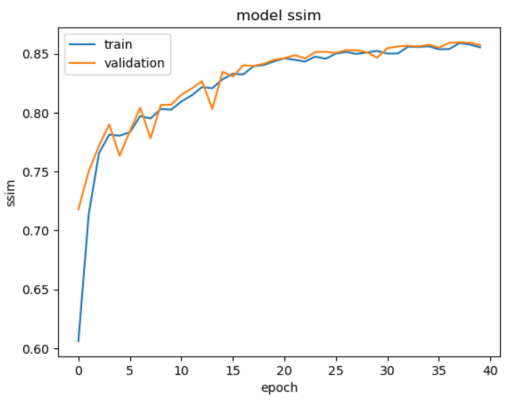
- *Single / Ordinary Autoencoder* - The below graphs are for loss (left), PSNR (middle) and SSIM (right) for a simple autoencoder with normal convolution layers. As it can be observed that the SSIM value peaks at about 0.825 and PSNR is just about 27.



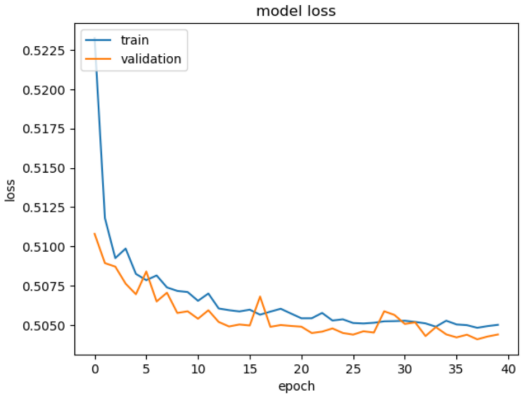
- *Dual Autoencoder* - The below graphs are for loss (left), PSNR (middle) and SSIM (right) for a dual autoencoder with normal convolution layers. As it can be observed that the PSNR just crosses 27 whereas SSIM is touching 0.85.

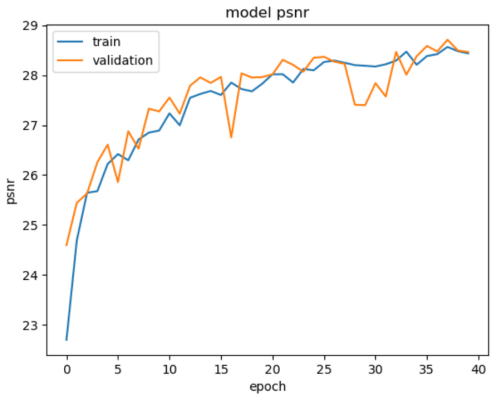


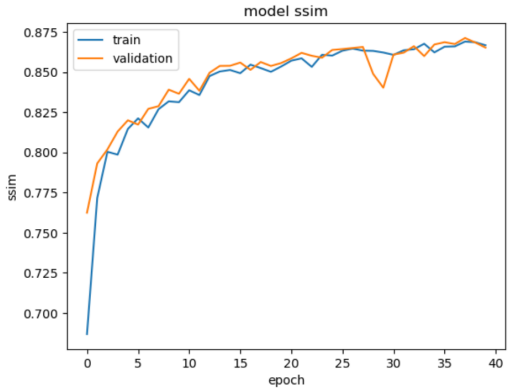




- *Dual Autoencoder with skip connections* - The below graphs are for loss (left), PSNR (middle) and SSIM (right) for a dual autoencoder with normal convolution layers and skip connections. As it can be observed that the SSIM value is almost 0.875 whereas the PSNR is closing in on 29.







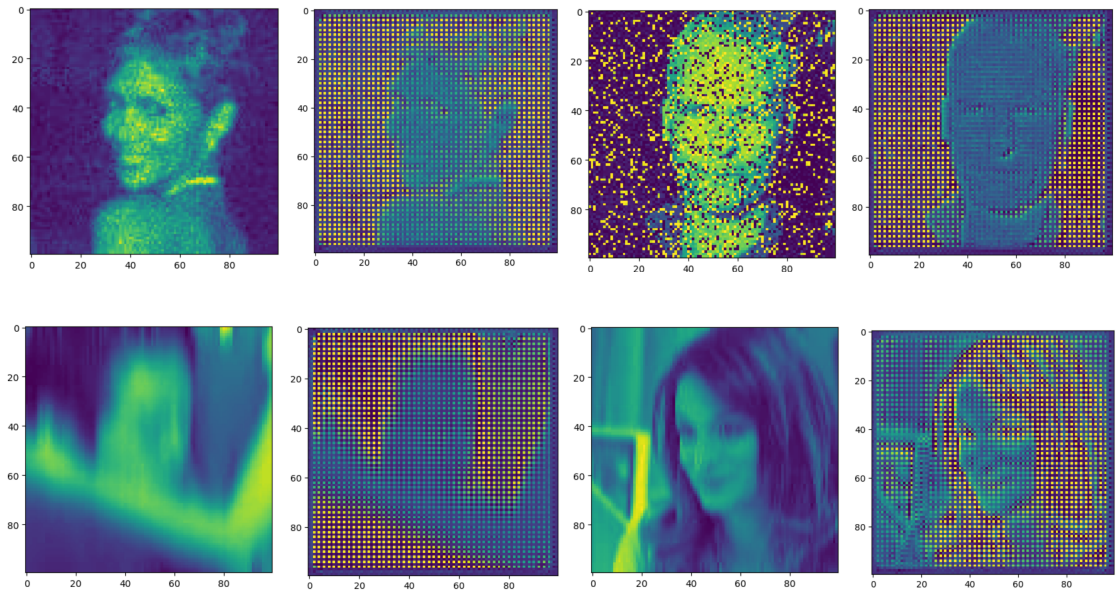
For the IMDB-WIKI dataset the ordinary autoencoder gave the least PSNR and SSIM values of around 27 and 0.825 followed by the dual autoencoder with the PSNR value just over 27 and SSIM of about 0.85. Finally, the dual autoencoder with skip connections gave us the best results with PSNR around 29 and SSIM of around 0.875. When it comes to parameters, as expected the dual autoencoder with skip connections has the highest number of parameters but it is very close to what a normal dual autoencoder but it gives a significantly better performance. The overall performance of all the models is directly proportional to the number of parameters it has. Now, we also tried out the separable convolution versions of the three models described above, some of them gave comparable results with significantly lower numbers of parameters. *For the scope of this report and keeping in mind the restrictions we will not be discussing them in detail here*. However you can find a detailed table below that contains all the models with their variants, their performance scores and the number of trainable parameters they use. As we can

observe, based on the type of noise we have evaluated all the models on the test set and *for all the types of noise but one Dual Autoencoder with skip connections works best. Whereas for the gaussian noise Dual Autoencoder without skip connections is the best with a slightly higher score.*

Model Name	Noise Type	PSNR	SSIM	Trainable Parameters
Single Autoencoder	Gaussian	24.30	0.76	36,386
	Impulse	25.99	0.82	
	Speckle	27.19	0.86	
	Poisson	28.14	0.89	
Single Autoencoder (Separable Convolutions)	Gaussian	24.18	0.75	15,760
	Impulse	22.12	0.62	
	Speckle	26.78	0.83	
	Poisson	27.51	0.87	
Dual Autoencoder	**Gaussian**	25.01	0.78	87,812
	Impulse	26.88	0.85	
	Speckle	28.11	0.88	
	Poisson	29.08	0.90	
Dual Autoencoder (Separable Convolutions)	Gaussian	24.65	0.76	52,445
	Impulse	22.79	0.67	
	Speckle	27.26	0.86	
	Poisson	27.91	0.88	
Dual Autoencoder with skip Connections	Gaussian	24.92	0.76	91,488
	Impulse	28.59	0.88	
	Speckle	29.52	0.89	
	Poisson	30.83	0.91	
Dual Autoencoder with skip Connections (Separable Convolutions)	Gaussian	24.37	0.75	56,231
	Impulse	24.27	0.73	
	Speckle	27.95	0.87	
	Poisson	28.84	0.89	

Below are the few sample outputs from the best models for a particular type of noise based on the table above. *The best models in the above table for a particular noise type are highlighted in BOLD.* All the four noise types with their outputs are shown below where noise image (left), clean image (middle) and denoised image (right).

In the images below you can see the intermediate activations for the dual autoencoder model on the final trained model. The activations are taken for the middle layer i.e. the output of the first autoencoder, its dimensions are (100, 100, 1). As we can clearly see that it is able to successfully remove the noise from the image by that stage as it is completely focusing on the image facial features and not on the noisy pixels. The mask generated by it is capable of giving us a noise-reduced image on its own even if we ignore the second autoencoder after it. This shows that the first autoencoder is aiming for the noise reduction while the second focuses on enhancing the image reconstruction quality.



References

[1] Dual Autoencoder Network with Separable Convolutional Layers for Denoising and Deblurring Images by Elena Solovyeva and Ali Abdullah published on 13th September 2022 by MDPI
https://www.mdpi.com/2313-433X/8/9/250

[2] Deep Residual Learning for Image Recognition by Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun and Microsoft Research
https://arxiv.org/pdf/1512.03385.pdf

[3] MobileNets : Efficient Convolutional Neural Networks for mobile vision applications by Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, Hartwig Adam
https://arxiv.org/pdf/1704.04861.pdf

[4] IMDB-WIKI Dataset
https://data.vision.ee.ethz.ch/cvl/rrothe/imdb-wiki/

[5] FER-2013 Dataset:
https://www.kaggle.com/datasets/msambare/fer2013

[6] KDEF Dataset:
https://kdef.se/

[7] Affectnet Dataset:
http://mohammadmahoor.com/affectnet/