Tackling Blur due to Camera Motion

Abhilash Pravin Mane, Pradnya Mundargi, Tanuj Thakkar

Abstract—**Blur caused by motion of a camera in a stationary scene is a challenging computer vision as well as computational imaging problem. In this project, we attempt to resolve this problem using various approaches based on well known architectures such as U-Nets, ResNets and GANs. Our six different models attempt to tackle the deblur by using various techniques such as having a multi-scale architecture, modified loss functions and the trade off between wide and deep networks. The results obtained shows that deep network as much more efficient in deblurring as compared to the wide networks. Also using loss functions depending upon the nature of the objective helps the models to converge fast and get high accuracy as compared to the general L2 loss.**

1. Introduction

Recently, drones have gained popularity for a large spectrum of application from delivery to rescue. For applications in which, time is of the essence, drones perform very well given their agility and aggressive flights. However, the drawback in these situations is that at high speeds, the video feed from the camera attached on the drone gets blurred due to its fast motion or vibrational disturbances. This puts an cap on the speed limit of the drone. A major portion of the previous work has focused on solving blur caused by translation and rotational motion. With the advancement of deep learning in computer vision and computational imaging, the problem of deblur is generalized and tackled as an uniform blur. Debluring can be seen as an inverse problem where a sharp image is blurred using a blur kernel.

$$O = BS + n \tag{1}$$

Here, O corresponds to the blur image, S is the vectorized sharp image, B is a blur kernel and n to noise. Usually B and S are unknown. In deep learning method a model is trained on blur and sharp images to obtain an approximate blur kernel B (usually the weights) and during the inference time, S is predicted given the Blur kernel and Blur image, O. Recent efforts have shown attempts in training deblurring models using Convolution Neural Network (CNN) in a supervised learning method which require a ground truth (sharp image) and the corresponding input(blur image). However, obtaining a real world data is an difficult task which had led to the development of synthetic method of generating the dataset. Particularly for the the problem of blur-deblur, high FPS videos are captured and frames are averaged out to get an approximate blur image.

When the problem of deblurring is considered as a 3-D problem in the real world, in place of a 2-D problem, factors such as scene depth variation have an effect on the extent of the blur need to be taken into consideration. In order to tackle this problem, we have implemented multi-scale, multi-loss as well as deep and wide neural networks which may be capable of detecting features at various intensities of blur. For the purpose of our model training and experiments, we have used the dataset generated by a GOPRO4 Hero Black camera which captured 240 fps videos. The frames were then averaged by a varying number to create a dataset of blurred images.

2. Related Work

Deblurring has been one of the most widely researched problems in the field of computational imaging. Solving motion blur due to shakes of camera and fast object motions has particularly garnered wide attention from the research community not only from the computational imaging field but also from other fields like robotics. Obtaining sharp estimates of motion blurred scenes in an important aspect of problems in the field of robotics as blur induces uncertainty in data which in turn decreases the efficacy of the visual algorithms.

The recent works of literature have been using various forms of Convolutional Neural Networks (CNNs) to solve the inverse problem of obtaining sharp estimates from blurry inputs. In [1], Generative Adversarial Network (GAN) is proposed, where the generator of the GAN is a CNN based on the ResNet architecture which produces sharp estimates of blurry inputs. There estimates are then distinguished by a discriminator which receives a real and a generated sample. The paper uses a combination of adversarial loss and typical $\mathcal{L}_1$ or $\mathcal{L}_2$ loss. In [2], a multi-scale CNN is proposed to directly produce sharp estimates, skipping the restricted blur kernel estimation step.

3. Our Approach

We have attempted to train and test six different motion deblurring models which have variations in terms of architecture as well as approach. Our models are based on existing state-of-the-art networks such as ResNets, Unets and GANs the details of which are provided in the following sections.

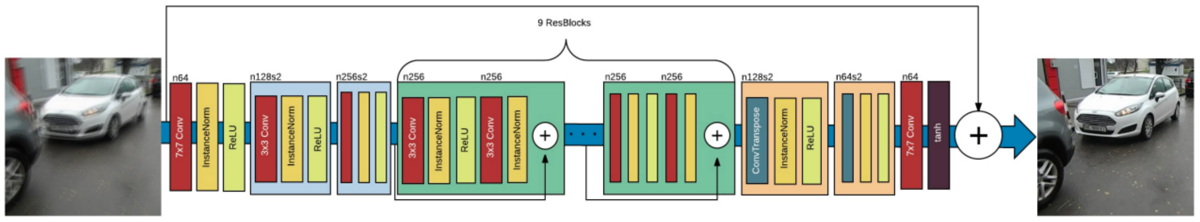

Figure 1: ResNet Architecture from [1]. The ResNet consists of two strided convolution blocks followed by nine ResBlocks, each with two convolution layers. Two blocks of transposed convolution follow the ResBlocks with a final convolution layer in the end

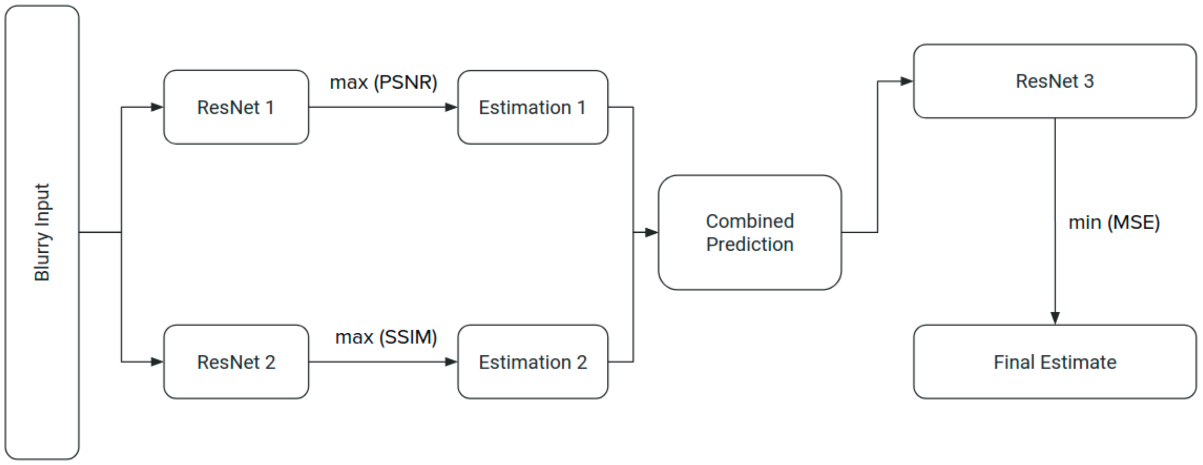

Figure 2: The Multi-ResNet Model

3.1. The ResNet Model

In this approach, we employ a CNN based on the ResNet architecture [3] to produce sharp estimates of blurry inputs. The idea stems from [1], where the generator of the GAN employed a ResNet based CNN to generate sharp estimates of blurry inputs which were then critiqued by the discriminator. In our approach however, we only utilize the ResNet based CNN without the discriminator to critique the generated estimates. The ResNet architecture shown in Fig. 1, is used which has two $\frac{1}{2}$ strided convolution blocks, followed by nine residual blocks (ResBlocks). Each ResBlock consists of two convolution layers, where after each convolution layer, there is an instance normalization layer, and ReLU activation layer. Between the two convolution layers in the ResBlock, dropout regularization is performed with a probability of 0.5. After all the ResBlocks, there are two blocks of transposed convolution followed by a final convolution layer. We use the $\mathcal{L}_2$ loss or the Mean Squared Error (MSE) for learning.

3.1.1. Training. The model was implemented in TensorFlow 2 [4] and trained on a single Nvidia GeForce RTX 2070 Super. As mentioned in Sec. 3, we used the GOPRO dataset for training our models. The original 1280×720 images were resized to 256×256. A batch size of two was used for training. The ADAM optimizer starting with a learning rate of 0.001 was employed with the learning rate being halved every 100 epochs. The network was trained for a 1000 epochs and obtained an average PSNR of 29.400, with an average SSIM of 0.870. More quantitative and qualitative results are shown in Table 1 and Fig. 6 respectively.

3.2. The Multi-ResNet Model

To extend on the results obtained from the single ResNet based CNN as described in Sub-Section 3.1, we implemented a Multi-ResNet Model. As the name suggests, the Multi-ResNet model consists of multiple, but identical ResNets which are commissioned to produce sharp estimates based on different metrics. The Multi-ResNet model is shown in Fig. 2. The Multi-ResNet framework can accommodate n number of ResNets to learn different parts of the inverse problem. In our particular implementation, we use three ResNets, where one network is trained to maximize PSNR, and another network is trained to maximize SSIM. The third and final ResNet, takes the combined estimates of the these networks, to produce the final sharp estimate. The final ResNet is again trained to minimize the $\mathcal{L}_2$ loss.

In this framework, the performance of the overall network vastly depends on the strategy that is used to combine the intermediate estimates. For the sake of simplicity, we use

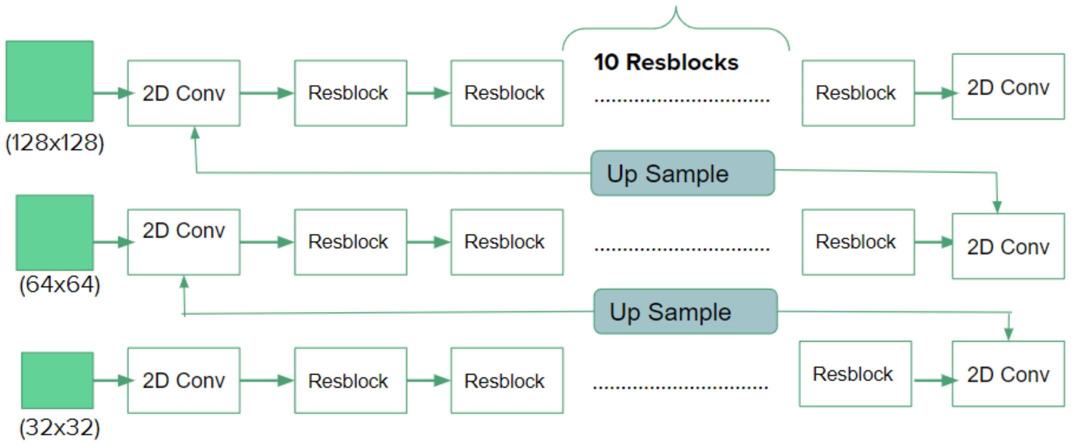

Figure 3: Resblock Architecture

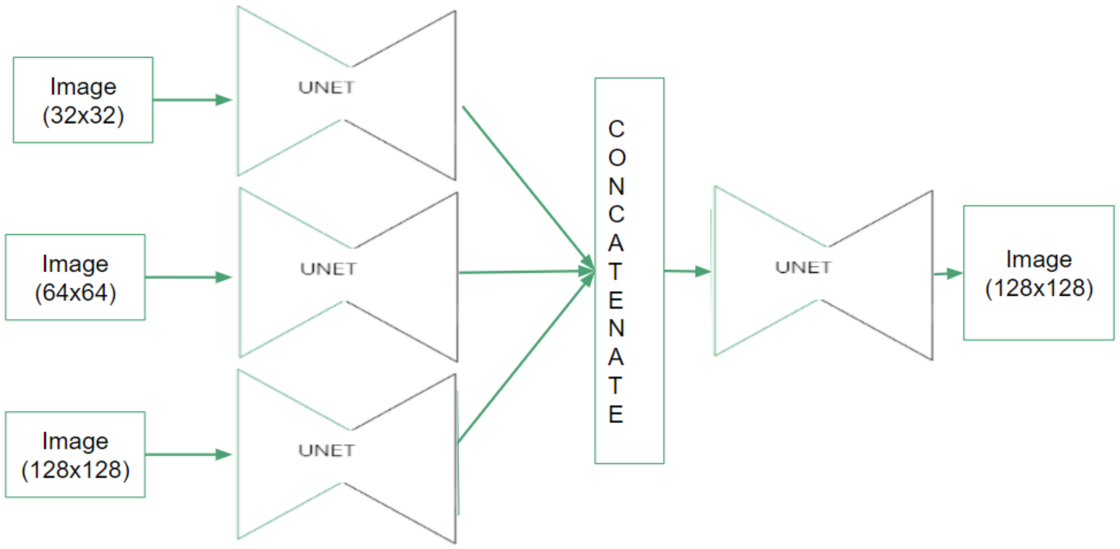

Figure 4: Xception style UNET Architecture

the average of the two intermediate estimates. However, one challenge here is that averaging two estimates will increase neighbourhood blur, therefore we use a typical sharpening filter to tackle the issue.

3.2.1. Training. Similar to (3.1.1), a batch size of two was used for training. The ADAM optimizer starting with a learning rate of 0.001 was employed with the learning rate being halved every 50 epochs. The network was trained for a 1000 epochs for which the model-wise quantitative results are detailed in Table 2. Qualitative results are shown in Fig. 6.

3.3. Deblur Using GAN Model

In accordance with the deblurring approach followed in [2], we have attempted to recreate the model. The model has the typical architecture of a GAN consisting of a generator and a discriminator. The generator, which is also the delurring model, has multiple finer to coarser layers, similar to a gaussian optimising network. Each scale consists of a convolution block, followed by 19 resblocks and another convolutional layer in the end. The last image in each layer is upsampled and concatenated with the input image at the next scale. The discrimminator has 8 convolutional layers, followed by a fully connected layer and gives an output of either ’blur’ or ’not blur’ using a sigmoid activation function. The activation function within each layer is the Leaky Relu.

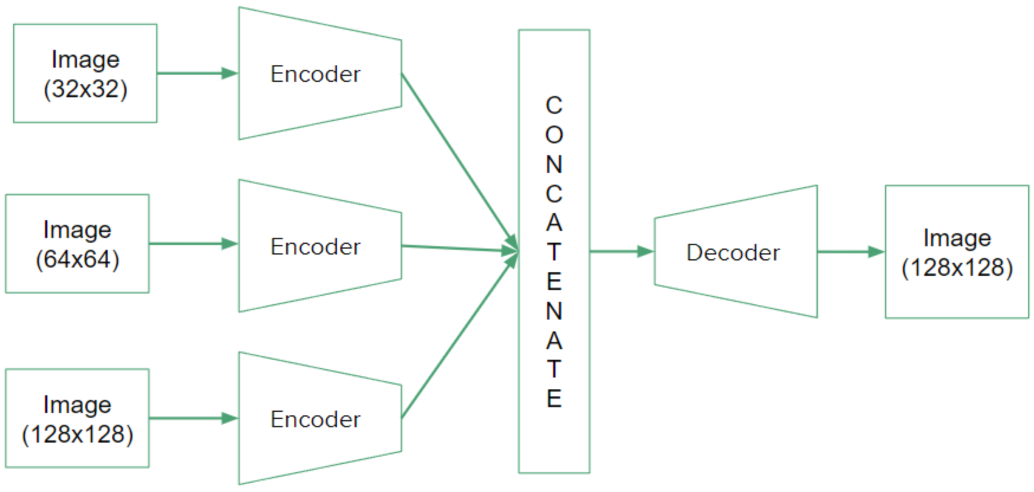

Figure 5: Xception style UNET Latent Vector Architecture

3.3.1. Training. The model was implemented in TensorFlow 2 [4] and is NVidia Tesla K80 GPU was used for training. Reffering to . 3, we used the GOPRO dataset for training our models.Being a multi scaling model, images were scaled to 124×124, 64×64,32×32 . Following were the parameters used for training: batch size of 16 , adam optimizer learning rate of 0.0001, epochs 500. Total loss which is combination of adverserial and L2 loss is used as a loss function.

3.4. Modified Reslock Model

Taking inspiration from [2], we attempted to modify the resblock structure as well as convert a GAN network into a simple supervised neural network model which would have a faster inference speed. The resblock, having a similar architecture to a resnet as shown in 3 has 10 repeating units between two convolution blocks at either end. We have remove the batch normalisation layers which subsequently increases the efficiency and convergence of the model. We also decreased the number resblocks by half which significantly decreases the runtime. We use PSNR as the loss function which increase the convergence and also increase the model accuracy. Receptive field can be increase by stacking the resblock but it hampers with the inference speed. So the number of reblock stacked should be optimum. We use 3 scale of in the order of finer to coarser. The model architecture consists of input blur images are of size (128,128,3),(64,64,3),(32,32,3) respectively a convolution layer followed by the 10 Resblocks which is attached with a convolution layer at the end which outputs a sharp image of size (128,128,3).

3.4.1. Training. The model was implemented in TensorFlow 2 [4] and is NVidia Tesla K80 GPU was used for training. Reffering to . 3, we used the GOPRO dataset for training our models.Being a multi scaling model, images were scaled to 124×124, 64×64,32×32 . Following were the parameters same as the above model are used for training with PSNR as a loss function.

3.5. Multi-scale Xception style UNET Model

The model has acquired most of its architecture inspiration from the XCEPTION model which is based on depthwise separable convolution layers [5]. This model contains a multi-scale architecture of three parallel Xception style Unets whose outputs are concatenated and serve as an input to the fourth Xception style Unet as shown in 4. A single image is scaled to three different image sizes of (124,124,3), (64,64,3) and (32,32,3) and are provided as input to the first three Unets as highlighted. The output image size of all images from the three networks is converted to (123,124,3) by upsampling the lower sized images. This process is followed by concatenation of the three images resulting in an intermediate latent output of size (128,128,9) which serves as input to the fourth XCeption style Unet. The three initial networks in our model attempts to detect features at various blur intensities and the fourth network attempts to resolve the overall blur of the image.

3.5.1. Training. The model was implemented in TensorFlow 2 [4] and is trained with Tesla K80 GPU. As mentioned in Sec. 3, we used the GOPRO dataset for training our models. images were resized to 124×124, 64×64,32×32 for different scale. A batch size of 16 was used for training. We used adam optimizer along with learning rate of 0.0001 with decay of 0.05 after every epoch. The model is trained for 500 epochs on PSNR as loss function.

3.6. Multi-scale Xception style Unet Latent vector Model

This model focuses on deblurring as well as reducing inference time in the presence of a large number of parameters. The model architecture and approach is based on the network provided in [6]. We modified the above model to reduce the number of parameters to decrease the inference time. The assumption behind this is that most of the high quality data required for the deblurring is captured in a latent vector. Three different images of size (124,124,3), (64,64,3) and (32,32,3) are feed as an input to 3 different xception style classifying model as seen in 5.These models are modified such that they don't have any layers from flattening and hence the output of these layers is a latent vector . Before concatenating, output of all these latent vectors are scaled to the same dimension by doing extra convolutions for images with large size. These latent outputs are then concatenated together so that we get the high quality data from each scale. Once concatenated vectors are obtained, they are passed through a deconvolution model similar to the deconvolution part of the Xception style UNET model. This deconvolution model tries to map the hiqh quality data obtained from the 3 convolution models to output a sharp image of size (128,128,3).

3.6.1. Training. The model was implemented in TensorFlow 2 [4] and is trained with Tesla K80 GPU. Reffering to . 3, we used the GOPRO dataset for training our models. Images were scaled to 124×124, 64×64,32×32 .A batch size of 16 , adam optimizer along with learning rate of 0.0001 with decay of 0.05 after every epoch was used for training the model. The model is trained for 500 epochs on PSNR as loss function.

4. Results & Analysis

The quantitative results of all the models implements are detailed in Tab. 1. The Single ResNet seems to outperform other models which is surprising as it is the most simple model of the bunch. The performance of Multi-ResNet is inferior to that of the single ResNet in terms of PSNR as well as SSIM. However, the combined prediction from the two intermediate networks seems to perform better.

As can be seen from the result table, the deeper neural networks work better than the wider custom models. Adding or removing Batch Normalization does create a difference in the model performance. For the Modified Resblock, Multi-scale Xception U-Net and Multi-scale Latent Vector models, the negative value of PSNR is used as a loss function which the models have attempted to reduce.

5. Conclusion

In conclusion our project attempts to recreate as well as design our own custom models for the purpose of debluring caused by camera motion. Among the designed models, a few show promising results.

TABLE 1: Quantitative Results

Model	**Avg. $\mathcal{L}_2$**	**Avg. PSNR**	**Avg. SSIM**
Single ResNet	0.006	29.400	0.870
Multi-ResNet	0.013	25.240	0.758
Deblur Using GAN Model	-	23.10	
Modified Resblock Model	-	25.26	0.84
M.S*.Xception style U-NET	-	17.23	0.61
M.S.* Latent vector	-	15.29	0.43

* denotes Multi Scale

TABLE 2: Multi-ResNet Model-wise Results

Model	**Avg. $\mathcal{L}_2$**	**Avg. PSNR**	**Avg. SSIM**
ResNet 1	0.007	28.840	0.854
ResNet 2	0.010	27.485	0.849
Combined Pred.	0.006	30.015	0.887
ResNet 3	0.013	25.240	0.758

6. Contributions

1) Dataset availablity - Abhilash Mane
2) ResNet Model - Tanuj Thakkar
3) The Multi Resnet Model - Tanuj Thakkar
4) Deblur Using GAN Model - Pradnya Mundargi
5) Modified Resblock Model - Pradnya Mundargi
6) Multi-scale Xception style U-NET Model - Abhilash Pravin Mane
7) Multi-scale Xception style UNET Latent vector Model - Abhilash Pravin Mane

References

[1] O. Kupyn, V. Budzan, M. Mykhailych, D. Mishkin, and J. Matas, "Deblurgan: Blind motion deblurring using conditional adversarial networks," *ArXiv e-prints*, 2017.

[2] S. Nah, T. Hyun Kim, and K. Mu Lee, "Deep multi-scale convolutional neural network for dynamic scene deblurring," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 3883–3891.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[4] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[5] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1251–1258.

[6] R. Martin-Brualla, N. Radwan, M. S. Sajjadi, J. T. Barron, A. Dosovitskiy, and D. Duckworth, "Nerf in the wild: Neural radiance fields for unconstrained photo collections," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 7210–7219.

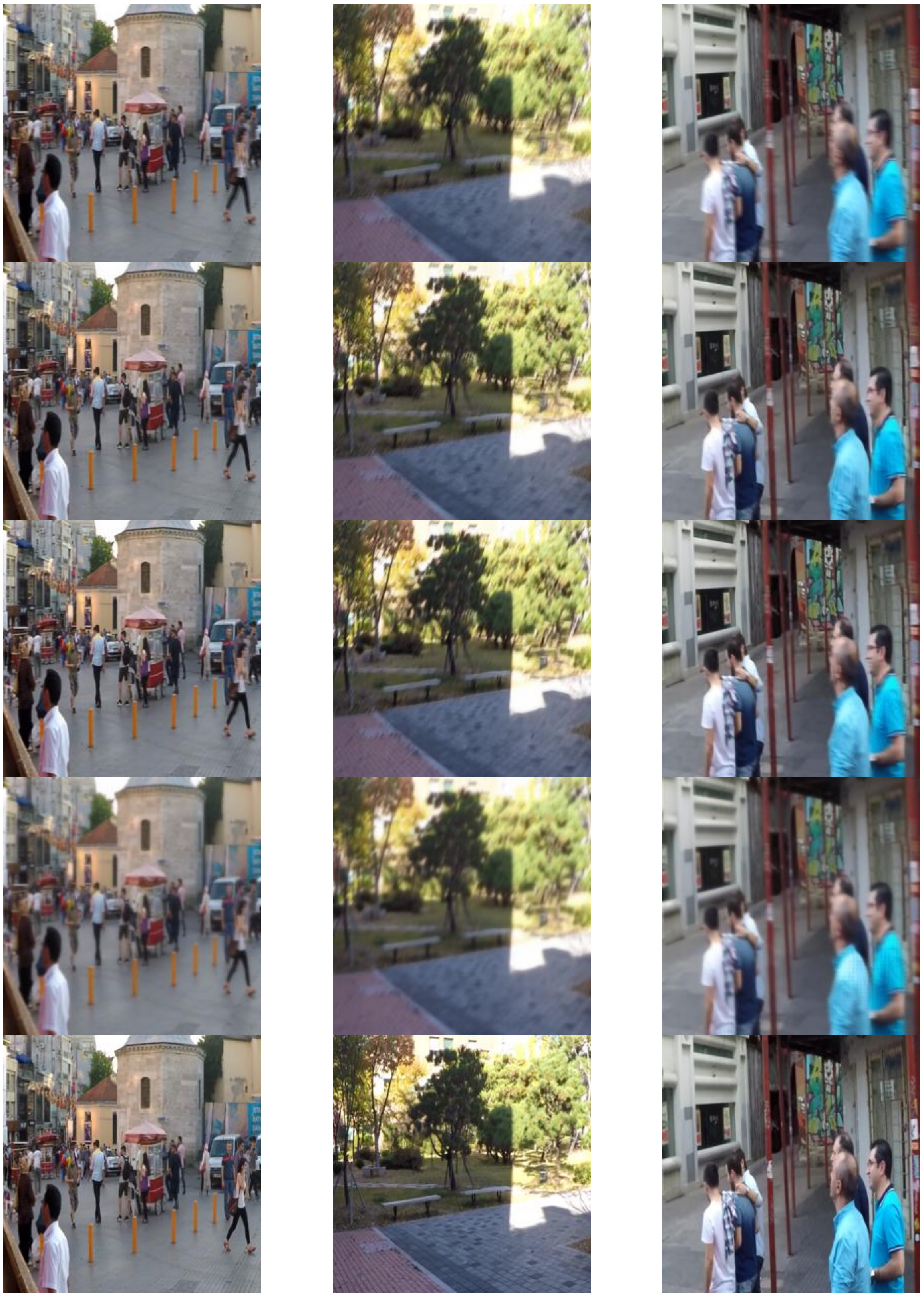
Figure 6: Qualitative Results for Single ResNet and Multi-ResNet Models
Vertical Order - Blurry Input, Single ResNet, Combined Prediction of ResNet-1 & ResNet-2 from Multi-ResNet, Multi-ResNet, Ground Truth

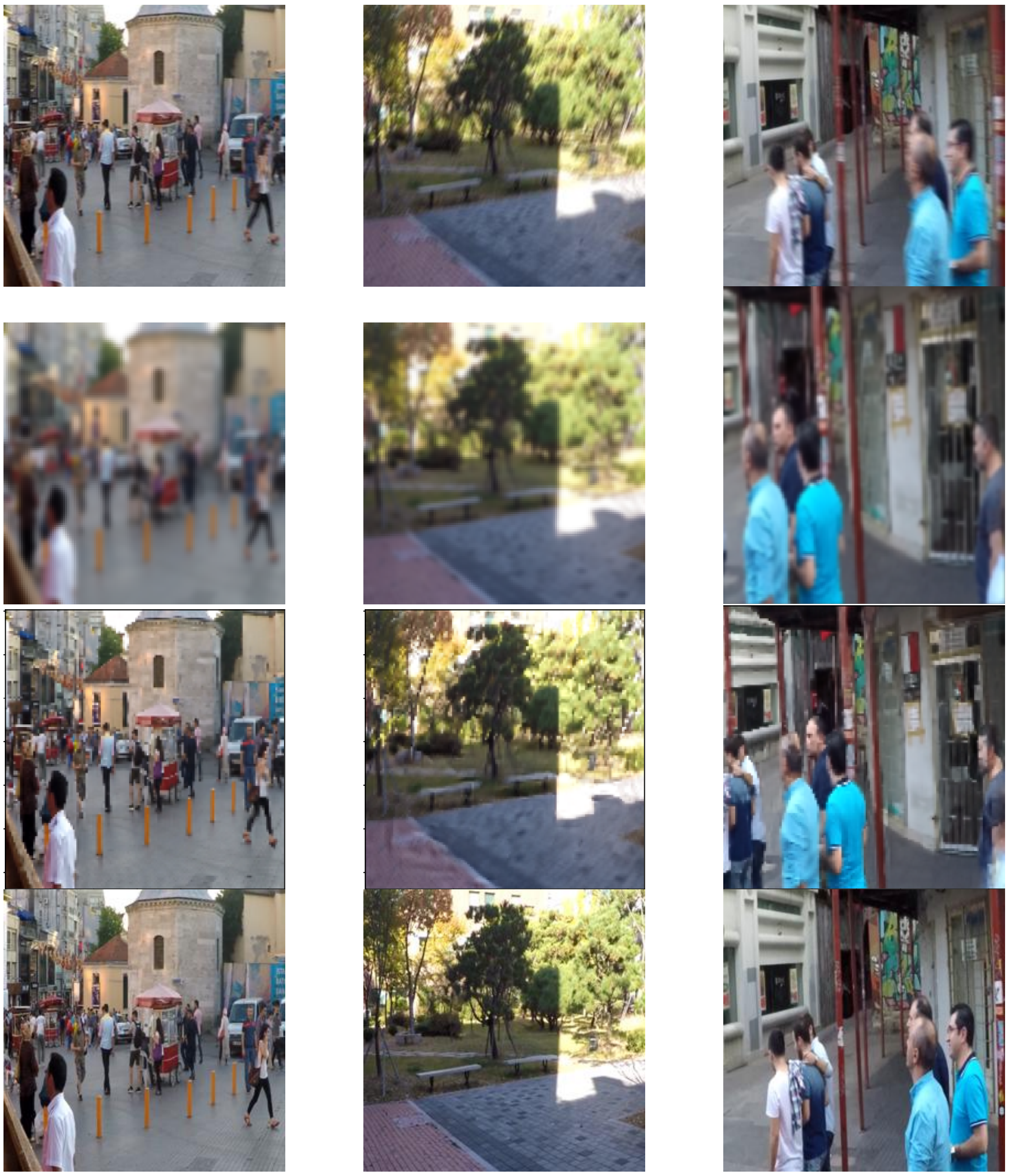

Figure 7: Qualitative Results for GAN & Modified ResBlock Model
Vertical Order - Blurry Input, GAN, Modified ResBlock Model, Ground Truth

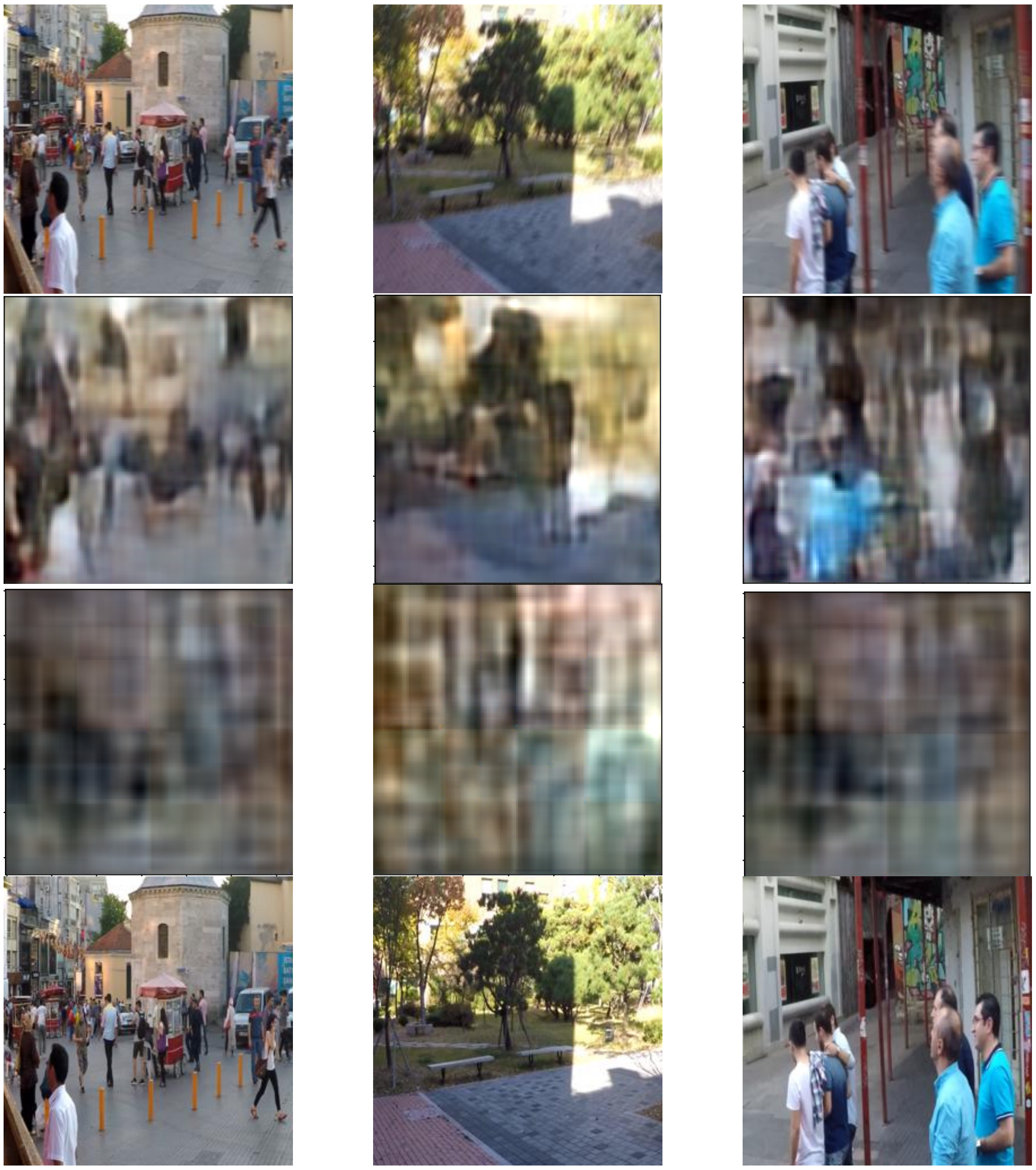
Figure 8: Qualitative Results for Xception Style U-Net & Xception Style U-Net Latent Vector Model
Vertical Order - Blurry Input, Xception Style U-Net, Xception Style U-Net Latent Vector Model, Ground Truth