CSE251B Project: Recognize Blurry Image with Transfer Learning

Chen Shen c3shen@ucsd.edu
Zunding Huang zuhuang@ucsd.edu
Futian Zhang f6zhang@ucsd.edu
Weihong Xu wexu@ucsd.edu

Abstract

In this paper, we develop a deep learning model to restore blurry images to improve the performance of classification on blurry images. We train our restoration model on a large dataset and apply transfer learning to restore blurry images on other datasets. Our best performing restoration model is a modified UNet. It can successfully restore the blurry image and increases the accuracy performance of a simple CNN model by 5.13%, from an average of 87.39% to 92.52% across all test datasets.

1 Introduction

1.1 Problem Definition

The training goal of image classification is to estimate the model weights θ for the classification model $f(\boldsymbol{x};\theta)$ as follows:

$$\theta = \underset{\theta}{arg\,min}\ \mathcal{L}\left(f(\boldsymbol{x};\theta), \boldsymbol{y}\right) + \mathcal{R}(\theta), \tag{1}$$

where the image and label pairs $(\boldsymbol{x}, \boldsymbol{y})$ belong to the training dataset. $\mathcal{L}(\cdot)$ denotes the loss function while $\mathcal{R}(\cdot)$ represents the regularization term imposed on θ. CNN is often used to perform tasks with image with high resolution. Most datasets remove the blurry images and assume that the input $\boldsymbol{x}$ does not contain blurry images because they degradate the accuracy.

Figure 1: Illustration of blurry images.

However, the image blur has huge impact on the final classification accuracy. They occur due to the following factors: 1. Camera takes time to focus and the lens out-of-focus may produce blurry images; 2. Camera shake or fast moving objects may cause blur effects;

3. Last but not least, scaling up images when the original resolution is low. As shown in Figure 1, the blurry objects in the example images are harder for the classification model to recognize. Hence, it is important and meaningful to study the performance of CNN on blurry images because in reality, blurry images are inevitable and common in photos. However, few work considers the impact of blur on the performance. Our group is curious about the performance of CNN on blurry images and how to improve the performance.

We reformulate the image classification problem in Eq. (1) after considering the blur effects $h(\cdot)$ imposed on the original images. The problem definition is re-written as:

$$\theta = \underset{\theta}{arg\,min}\ \mathcal{L}\left(g(h(\boldsymbol{x});\theta), \boldsymbol{y}\right) + \mathcal{R}(\theta), \tag{2}$$

where $g(\cdot)$ denotes the proposed recognition algorithm in this project. Instead of classifying the original images $\boldsymbol{x}$, the input for the classifier $g(\cdot)$ are blurry images represented by $h(\boldsymbol{x})$ and the output is the label of the object. The goal is to achieve the best performance in terms of validation accuracy and loss.

1.2 Overview

In this paper, we use transfer learning to restore the blurry images before classification. We first build a classification model using simple CNN, and we measure the performance of it on our datasets. Then we use transfer learning to make the blurry images clear. We trained a second neural network called Restore CNN. Restore CNN takes a blurry image as input and output a clear image. The structure of Restore CNN is a modified UNet. Restore CNN is trained on a large datasets, then it is used as a pretrained model to preprocess the blurry images for other datasets. After preprocessing, the images are trained and tested on our simple CNN. We measure the performance on our preprocessed datasets and compare with the blurry datasets.

For the purpose of fast training/testing, we use datasets with small, grayscale images. We apply our method on Fashion-MNIST, USPS and Extended MNIST dataset with a blurry filter. We train our Restore CNN on EMNIST and use it to preprocess all datasets to determine if the performance of classification model improved with Restore CNN.

2 Related Work

Many publications focus on restoring blurry images and classification task on small images.

For restoring blurry images task, there are some famous papers like [1] and [7]. In [7], they present a neural network algorithm to directly restore a clear high-resolution image from a blurry low-resolution input. This motivates us to use neural network to solve the task of restoring blurry images.

The authors in [9] presents a unified framework to estimate the spatially-varying blur map. First, the proposed framework, ABC-FuseNet, detects whether blur exists from high-quality blur response maps generated by a dilated fully convolutional neural network with pyramid pooling and boundary refinement layers. In the second step, the model distills high-level semantics and learns an attention map to adaptively localize the important content in the image if the blur exists. The final classification is performed on the fused maps of generated blur map, attention map, and sematic map. The experiment results show a performance improvement over the previous works.

For classification task on small images, we have Fashion-MNIST, USPS and Extended MNIST datasets.

The best method for Fashion-MNIST now is from Muhammad's work in [6]. The paper offers a good trade-off between the number of parameters and classification accuracy and improves the top-1 accuracy on Fashion-MNIST by 0.56%, compared to the state-of-the-art approaches. The best method for Extended MNIST now is from Kabir's work in [4]. In the proposed SpinalNet, each layer is split into three splits: input split, intermediate split, and output split. In Extended MNIST dataset, they achieve 95.88% accuracy.

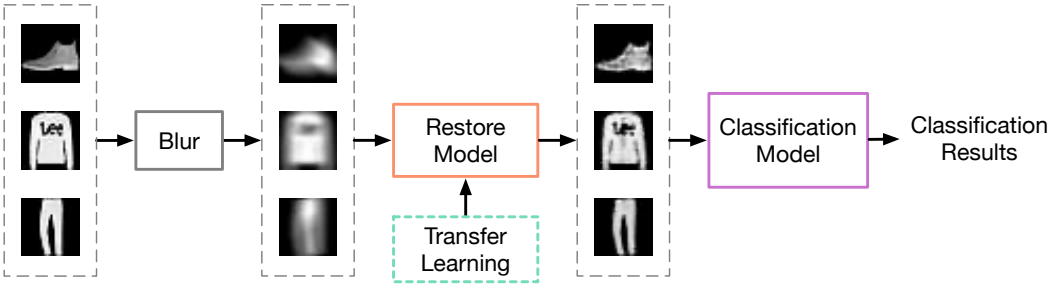

Figure 2: Overall pipeline of the blurry image recognition model.

3 Methods

3.1 Dataset

The dataset we use include Fashion-MNIST, USPS and Extended MNIST. We select these dataset because they contains small images, which is easy to train and manipulate.

Fashion-MNIST is a dataset comprising of 28×28 grayscale images of 70,000 fashion products from 10 categories, with 7,000 images per category. The training set has 60,000 images and the test set has 10,000 images. Fashion-MNIST shares the same image size, data format and the structure of training and testing splits with the original MNIST. We can easily get it from the following link.
`https://github.com/zalandoresearch/fashion-mnist`

USPS is a digit dataset automatically scanned from envelopes by the U.S. Postal Service containing a total of 9,298 16×16 pixel grayscale samples; the images are centered, normalized and show a broad range of font styles. We can easily get it from the following link.
`https://www.kaggle.com/bistaumanga/usps-dataset`

The EMNIST dataset is a set of handwritten character digits derived from the NIST Special Database 19 and converted to a 28x28 pixel image format and dataset structure that directly matches the MNIST dataset. We can easily get it from the following link.
`https://www.kaggle.com/crawford/emnist`

3.2 Overview

Figure 2 illustrates the overall flow of our studied blurry image recognition pipeline. In the first step, the original images are pre-processed by the `Blur` transformation module to emulate various types of image blurry effects. In the second phase, the blurry images are passed through a `Restore Model` to reduce the blurry impacts, thus reconstructing the original images for better classification. After the blurry images are restored, the `Classification Model` accepts the restored images and perform the classification on them. Besides, we also study the possibilities of using transfer learning for the `Restore Model`.

3.3 Blur Modeling

We apply two different blur transformations to the dataset, including Gaussian Blur transformation and Move Blur transformation.

3.3.1 Gaussian Blur

The first one is Gaussian Blur transformation. Gaussian Blur transformation can be seen as applying a mandatory convolutional layer, or filter, to the image. The filter has values resemble gaussian distribution, in which the center of the filter has the highest value and the value decreases as it gets closer to the edges. The value in each position can be calculated using:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x-c)^2+(y-c)^2}{\sigma^2}}$$

in which x, y is the position of the pixel, c is the center of the filter, σ is the variance. The filter is normalized to have sum 1. The Gaussian Blur transformation has a kernel size, which represents the side length of the filter. For this paper, we set the kernel size as 9 and variance to be 2. Notice that a kernel size of 9 is relative large since the size of the images are small.

3.3.2 Move Blur

In addition, we also generate a set of blurry data using Move Blur transformation. We consider that the moving camera often generate blurry images, therefore, we want to create a motion blur filter for our images. The motion blur filter is composed of a square with random box size, and only one diagonal (either the diagonal dexter and the diagonal sinister) has value of $\frac{1}{boxsize}$. We set the max box size to be 9. The filter is applied to the images after Gaussian Blur transformation. Therefore, these images are applied by two different filters.

Here are some example images in Fashion-MNIST after Gaussian Blur transformation and Move Blur transformation compared to original image:

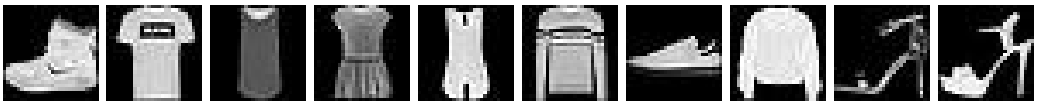

Figure 3: Original Images

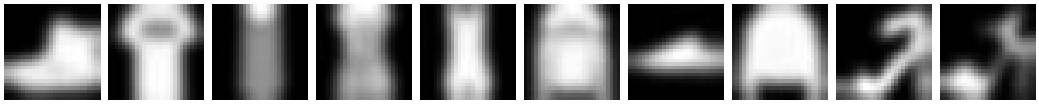

Figure 4: Images After Gaussian Blur Transformation

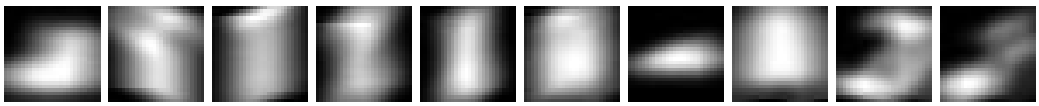

Figure 5: Images After Move Blur Transformation

3.4 Classification Model

Our model uses 3 convolutional layers whose (in_channels, out_channels, kernel_size, stride, padding) size are (1, 16, 5, 1, 2), (16, 32, 5, 1, 2) and (32, 64, 5, 1, 2) . Also, we use batch normalization, maxpooling and ReLU as activation function after each convolutional layer. Then, it is connected to a fully connected linear layer with softmax to classify. Without Restore CNN, this serves as our baseline model and we measure the performance of it against the images with Restore CNN.

3.5 Restore Model

Restore CNN is to restore the blurry images into original images. For Restore CNN, we have tried a lot of models and we find two models that performs well for this problem. We present these two kinds of models in our paper.

- **UNet**
  - The structure we use is only 2 down-sampling and 2 up-sampling block because of the image size is small.
  - For each down-sampling block, it consists of two convolution neural network with batch normalized and ReLU activation function following each cnn. One CNN doubles the channel number and one CNN halves the image width/height.
  - For each up-sampling block, it consists of one convolution transpose layer and a convolution neural network, batch normalized and ReLU activation function.
  - The kernel size is 9 instead of 3 in the original UNet architecture because we find that using a bigger kernel size can improve the performance of the model by including more pixels to determine the local features.
  - The output is one number per pixel, representing the converted pixel. Thus, the loss can be calculated by the mean squared error between the output of the model and target original clear image.
- **reverse-UNet**
  - We want to fix the above U-Net block is small and the network is a little bit shallow.
  - In this structrue, we firstly use up-sampling to enrich our pictures, and then use down-sampling to extract features.
  - The block number is not limited by the size of the image.

The architecture of these two models are shown below. The parameters are in channels (Cin), out channels (Cout), Kernal size (K), stride size (S), padding size (P), dilation size (D) and output padding size (OP) respectably. BN stands for batch normalization.

Layer	**Parameter (Cin, Cout, K, S, P, D, OP)**
Conv1+ReLU	$1, 64, 9 \times 9, 1, 4, 1, 0$
ConvEnc1+ReLU+BN	$64, 64, 9 \times 9, 1, 4, 1, 0$
Conv2+ReLU	$64, 128, 9 \times 9, 1, 4, 1, 0$
ConvEnc2+ReLU+BN	$128, 128, 9 \times 9, 1, 4, 1, 0$
Conv3+ReLU	$128, 256, 9 \times 9, 1, 4, 1, 0$
Tconv1+ReLU+BN	$256, 128, 9 \times 9, 1, 4, 1, 1$
Conv4+ReLU	$256, 128, 9 \times 9, 1, 4, 1, 0$
Tconv2+ReLU+BN	$128, 64, 9 \times 9, 1, 4, 1, 1$
Conv5+ReLU	$128, 64, 9 \times 9, 1, 4, 1, 0$
Conv6+ReLU	$64, 1, 9 \times 9, 1, 4, 1, 0$

Table 1: Network architecture for U-Net model

Layer	Parameter (Cin, Cout, K, S, P, D, OP)
Conv1+ReLU+Upsample	$1, 32, 3 \times 3, 1, 1, 1, 0$
Conv2+ReLU+Upsample	$32, 64, 3 \times 3, 1, 1, 1, 0$
Conv3+ReLU+Upsample	$64, 128, 3 \times 3, 1, 1, 1, 0$
Conv4+ReLU+Maxpool	$128, 64, 3 \times 3, 1, 1, 1, 0$
Conv5+ReLU+Maxpool	$64, 32, 3 \times 3, 1, 1, 1, 0$
Conv6+ReLU+Maxpool	$32, 1, 3 \times 3, 1, 1, 1, 0$

Table 2: Network architecture for Reverse U-Net model

3.6 Transfer Learning

Since our goal is to restore blur images, we assume that we are not presented with clear images. Thus, we can use transfer learning to obtain clear images without the present of training examples. We train Restore CNN on one of the datasets, EMNIST. The input of Restore CNN is the blur images and the target is the original clear images. We train Restore CNN so that it can recover the original clear images while presenting it with the blur images. Then Restore CNN is fixed and we use it to preprocess the images before classification for other datasets.

The reason why we choose EMNIST to train our Restore CNN for transfer learning is that there are 814,255 characters in EMNIST, which consists of 47 classes after merging. Hence it will provide enough data for us to train the Restore CNN.

For other datasets, the goal is to perform classification task on blurry images. The input is the blurry images and then we use Restore CNN to get the predicted clear images. For the classification CNN, we will input the both blurry images and predicted clear images, then we use our classification CNN model to train and test on the accuracy of labeling and to see if the performance has improved after we use the Restore CNN.

3.7 Training Details

3.7.1 Hyperparameter Configurations

In Restore CNN, we use MSE loss function and Adam minimizer to minimize the Mean-squared Loss function. The model is initialized with Xavier method. For our best results, hyper-parameters are as follows:

- The best learning rate in our experiments is 3×10^{-4}.
- Batch size is 64.
- Kernel size is 9×9, which means that to maintain the same output size, the padding is 4.
- Input images is randomly rotated from -180 to 180 degrees.

For Classification model, we use cross entropy loss function and Adam minimizer to minimize the loss function. The model is initialized with Xavier method. For our best results, hyper-parameters are as follows:

- The best learning rate in our experiments is 1×10^{-4}.
- Batch size is 64.
- Out channels each layer are 32, 64 and 128 respectively.

For both models, we use early stopping if the loss on validation does not decrease for continually 5 epoches.

3.7.2 Measuring Metrics

We use two different metrics to measure the performance of Restore CNN model. The first one is MSE (Mean Square Error) loss. We compare the output image with the original images without blur filter and measure the MSE score. MSE is calculated by the mean of square difference for each pixel between the output and target images. This can determine if the processed blur images is similar to the original image.

The second metric is the accuracy score after training with Classification model. For comparison, we have four different training trials for each dataset. The first two trials consist of directly performing classification on dataset with two blur filter (Gaussian Blur and Move Blur). The other two trials consist of applying the Recover CNN on dataset with two blur filter before performing classification. We also train Restore CNN model on the same dataset instead of using transfer learning for comparison. The accuracy score after classification is used to measure the improvement of performance with Recover CNN and show the effect of transfer learning on blur images.

4 Results

Here is the our first measuring metric for our two models, MSE loss of recovered images compared to the original image without blur filter for all datasets. The column blurry represents the loss of original clear image and the blur image. The column Restore represent the loss of original clear image and the restored image using the model trained by the same dataset. The column Restore(Transfer) represents the loss of original clear image and the restored image using transfer learning trained by EMNIST dataset. Notice for EMNIST, Restored and Restored(Transfer) has the same value because transfer learning is trained by EMNIST dataset.

Dataset	**Blurry**	**Restored**	**Restored(Transfer)**
Fashion-MNIST(G)	1.33×10^{-3}	1.94×10^{-4}	8.78×10^{-5}
Fashion-MNIST(M)	3.09×10^{-3}	2.76×10^{-4}	1.36×10^{-3}
USPS(G)	9.39×10^{-4}	4.21×10^{-5}	8.02×10^{-4}
USPS(M)	4.15×10^{-3}	6.17×10^{-5}	1.48×10^{-3}
Extened MNIST(G)	1.07×10^{-3}	6.68×10^{-6}	6.68×10^{-6}
Extened MNIST(M)	4.42×10^{-3}	1.23×10^{-4}	1.23×10^{-4}

Table 3: UNet MSE loss of Blurry and Recovered Images Compared to Original Image for all Datasets

Dataset	**Blurry**	**Restored**	**Restored(Transfer)**
Fashion-MNIST(G)	1.33×10^{-3}	2.07×10^{-4}	4.71×10^{-4}
Fashion-MNIST(M)	3.09×10^{-3}	5.89×10^{-4}	7.91×10^{-4}
USPS(G)	9.39×10^{-4}	1.07×10^{-5}	9.93×10^{-5}
USPS(M)	4.15×10^{-3}	5.06×10^{-4}	2.75×10^{-4}
Extened MNIST(G)	1.07×10^{-3}	1.91×10^{-4}	1.91×10^{-4}
Extened MNIST(M)	4.42×10^{-3}	3.12×10^{-4}	3.12×10^{-4}

Table 4: Reversed-UNet MSE loss of Blurry and Recovered Images Compared to Original Image for all Datasets

As we can see, the loss of UNet is less than the loss of Reverse-UNet for most cases. In addition, the loss of transfer learning generally performs worse than the loss of the model trained by the same dataset.

Here is classification accuracy with and without Restore CNN. Baseline refers to directly perform classification CNN without Restore CNN, and Restore refers to perform classifica-

tion CNN with Restore CNN. (G) refers to use Gaussian Blur filter and (M) refers to use Move Blur filer.

Model	**Blurry**	**Restored**	**Restored(Transfer)**
Fashion-MNIST(G)	89.20%	89.51%	89.56%
Fashion-MNIST(M)	83.32%	85.31%	89.62%
USPS(G)	92.33%	93.57%	95.81%
USPS(M)	84.72%	88.76%	95.07%
Extended MNIST(G)	86.43%	87.13%	87.13%
Extended MNIST(M)	82.82%	87.23%	87.23%

Table 5: Classification Accuracy with/without Restore CNN of UNet for all Datasets.

Model	**Blurry**	**Restored**	**Restored(Transfer)**
Fashion-MNIST(G)	89.20%	89.28%	88.57%
Fashion-MNIST(M)	83.32%	83.89%	82.76%
USPS(G)	92.33%	93.17%	93.52%
USPS(M)	84.72%	85.85%	91.13%
Extended MNIST(G)	86.43%	86.91%	86.91%
Extended MNIST(M)	82.82%	86.32%	86.32%

Table 6: Classification Accuracy with/without Restore CNN of Reverse-UNet for all Datasets.

As we can see, the accuracy of UNet is higher than the accuracy of Reverse-UNet for all cases. In addition, the accuracy of transfer learning generally performs better than the accuracy of the model trained by the same dataset.

Thus, we find that using our modified UNet, the accuracy of classification model has improved greatly. Transfer learning increases the accuracy performance of a simple CNN model by 5.13%, from an average of 87.39% to 92.52% across all test datasets.

Here are example test sets of original image, images with blur filter and restored images by using U-Net. Notice that the model is only trained on the training set of EMNIST.

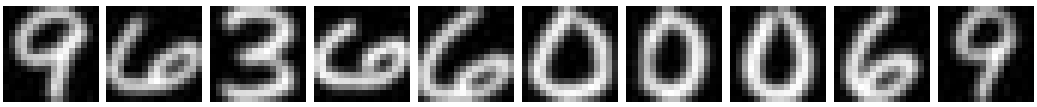

Figure 6: Original Images In USPS Dataset

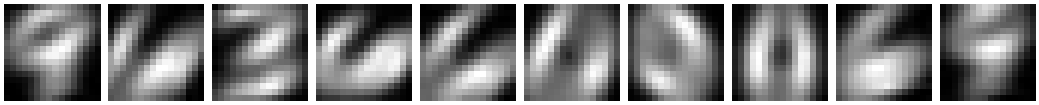

Figure 7: Images After Blur Transformation In USPS Dataset

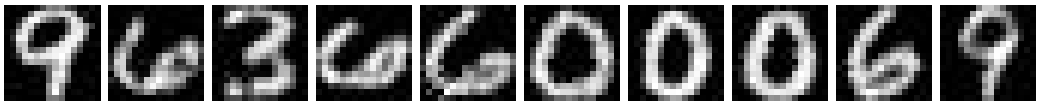

Figure 8: Restored Images In USPS Dataset

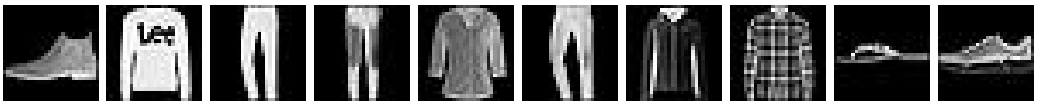

Figure 9: Original Images In Fashion-MNIST Dataset

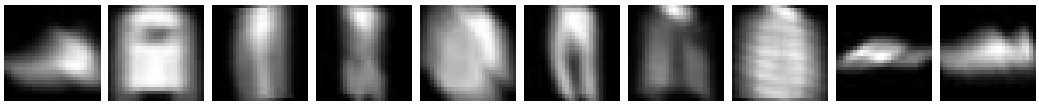

Figure 10: Images After Blur Transformation In Fashion-MNIST Dataset

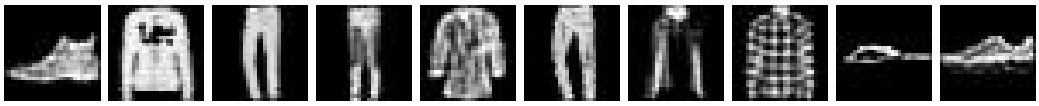

Figure 11: Restored Images In Fashion-MNIST Dataset

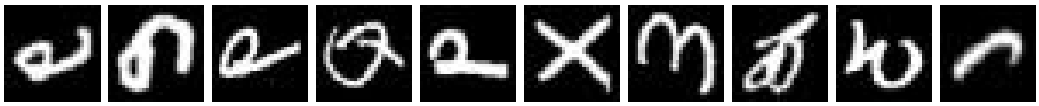

Figure 12: Original Images In EMNIST Dataset

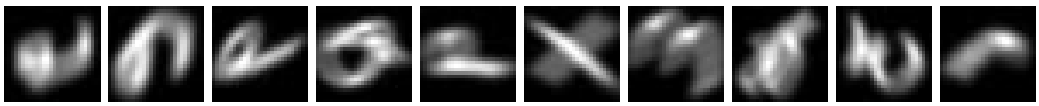

Figure 13: Images After Blur Transformation In EMNIST Dataset

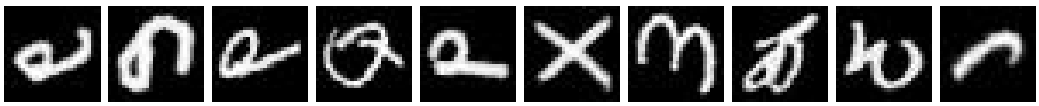

Figure 14: Restored Images In EMNIST Dataset

5 Discussion

There are 2 measuring metrics. For MSE loss Table, we can see that the loss for recovered images are much lower than the loss for blurry images, which means that our Restore CNN can clearly restore the blurry images, no matter the Gaussian Blur or the Move Blur. Also, the Restore CNN trained on Fashion-MNIST, and USPS returns lower loss than the Restore CNN for transfer learning trained on EMNIST. We believe that since our transfer learning model has not seen other datasets before, it is reasonable that it performs worse than the model that is trained on the same dataset.

For Classification Accuracy Table, it can be easily seen that in general, if we use Restore CNN, then the classification accuracy will improve a lot. This means that Restore CNN really work well if we use that for our datasets. We believe that transfer learning can perform so well because it is trained on such big dataset. Also, it can recover the details of the images, as we can see in above figures, it can even recover the letters on the shirt. However, it may not recover the image to the original location, which makes the loss high, but the accuracy of classification model performs well. To summary, our UNet Restore CNN restore blurry images quite well.

Also, it turns out that the Move blur method returns higher loss than the Gaussian blur method, which can also be seen from the formulas of these two blur methods. Move blur has a greater variance in the output images, which make restoring the images more challenging.

From example images, we can see that transfer learning quite play a good role in restoring blurry images. Even patterns in the original images can be recognized and then restored. Such as the letters on the shirt, the texture of the shoes.

Notice the accuracy for Transfer Learning Restore CNN is slightly higher than that for Restore CNN trained on Fashion-MNIST and USPS, respectively. We think the reason for this phenomenon is that compared with EMNIST, datasets Fashion-MNIST and USPS are much smaller. Then the Restore CNN is under-fitting when it is trained on Fashion-MNIST and USPS, respectively.

We believe that modified UNet performs well for a few reasons. First, it has a large receptive field. For Move blur, the pixel could be moved to a far location. A large receptive field can help obtain features from far away. Second, we modify the kernel size such that the more local feature can be read from each layer. Third, UNet has a similar structure as encoder and decoder, which allows the model to encode the blurry image with necessary features such that it can decode to the original image. Fourth, UNet has skip connections, which makes training easier and faster. Lastly, transfer learning is very useful in image tasks because many image tasks share similar approach.

6 Potential Applications

For our methods, there are many potential applications.

- Help police distinguish the suspect by restoring the low resolution CCTV videos.
- Recover old or damaged pictures.
- Distinguish the moving object in the picture.

7 Conclusion

In general, transfer learning can be used to restore blurry images. What's more, even patterns in the original images can be restored. More importantly, transfer learning of restoring blurry images can help CNN models to better recognize the images. Instead of kernel size of 3, it is better to use larger kernel size for the model that restores blurry images.

Further studies can elaborate on how the model performs on bigger or colored images. Also, for under-fitting, one good way for us to solve that is using more deeper networks and trained on a larger dataset.

Code

The code is available on `https://github.com/f6zhang/CSE251_Final`

Contribution

Chen Shen: Build and experiment different kinds of classification models; Build and experiment the reverse-UNet structure with different parameters; Write the training and testing process code; Write part of the report and presentation.

Zunding Huang: Build and Test the model on Fashion-MNIST and EMNIST for transfer learning respectively. Design and Experiment different UNet structures for the datasets. Write part of the report and presentation.

Futian Zhang: Write the blur filters, design and build UNet model structure, run and experiment with UNet and apply transfer learning to get the results. Write part of the report.

Weihong Xu: Prepare the code of adopted datasets. Test the performance of Transformer model on Fashion-MNIST. Write up part of the report.

References

[1] Sunghyun Cho, Yasuyuki Matsushita, and Seungyong Lee. Removing non-uniform motion blur from images. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8. IEEE, 2007.

[2] Zihang Dai, Hanxiao Liu, Quoc Le, and Mingxing Tan. Coatnet: Marrying convolution and attention for all data sizes. *Advances in Neural Information Processing Systems*, 34, 2021.

[3] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. *arXiv preprint arXiv:2010.01412*, 2020.

[4] HM Kabir, Moloud Abdar, Seyed Mohammad Jafar Jalali, Abbas Khosravi, Amir F Atiya, Saeid Nahavandi, and Dipti Srinivasan. Spinalnet: Deep neural network with gradual input. *arXiv preprint arXiv:2007.03347*, 2020.

[5] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021.

[6] Muhammad Suhaib Tanveer, Muhammad Umar Karim Khan, and Chong-Min Kyung. Fine-tuning DARTS for image classification. *CoRR*, abs/2006.09042, 2020.

[7] Xiangyu Xu, Deqing Sun, Jinshan Pan, Yujin Zhang, Hanspeter Pfister, and Ming-Hsuan Yang. Learning to super-resolve blurry face and text images. In *Proceedings of the IEEE international conference on computer vision*, pages 251–260, 2017.

[8] Abdou Youssef. Analysis and comparison of various image downsampling and upsampling methods. In *Proceedings DCC'98 Data Compression Conference (Cat. No. 98TB100225)*, page 583. IEEE, 1998.

[9] Shanghang Zhang, Xiaohui Shen, Zhe Lin, Radomír Měch, Joao P Costeira, and José MF Moura. Learning to understand image blur. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6586–6595, 2018.