EECS 453 Final Project Presentation

Joseph Lynch (lynchjo, 11442632)

Link to Recording:

https://youtu.be/T0vU4-qSvDE

Problem Definition

- Data augmentation is a powerful tool for image segmentation and classification tasks and can improve training efficacy and model generalizability[1]

- There are countless augmentation methods from simple geometric transformation to normalization and even intentionally adding noise

- In this project I sought out the answer to the following:

  "What data augmentation methods (or combination of methods) is most effective?"

Lecture 2, slide 92

UNIVERSITY OF MICHIGAN

Methods

Part 1: Create 10 different data augmentation techniques from scratch:

1. Horizontal Flip
2. Vertical Flip
3. Rotation
4. Perspective Shift
5. Crop
6. Crop and Paste
7. Gaussian Blur
8. Gaussian Noise
9. Adjust Sharpness
10. Adjust Brightness

Part 2: Write a UNET model to classify samples from the MNIST dataset that have been augmented

All work was done in python using NumPy and Random for creating the listed augmentation. PyTorch was used to create the model and dataloaders

Methods – Augmentations

Augmentation techniques were classified into two categories: geometric and pixel-based.

All augmentations were defined as methods with a "Compose" class that allows a user to apply any combination of the augmentations sequentially to an input image

Commentary:

- There was a wide range of difficulty in creating the different augmentations

```
import numpy as np
import random as rand

class Compose():
    def __init__(self, matrix, **kwargs):
        self.result = matrix
        available_funcs = ['flipHorizontal', 'flipVertical', 'rotate',
                           'perspectiveShift', 'cropResize', 'cropPaste',
                           'gaussianBlur', 'gaussianNoise', 'adjustSharpness',
                           'adjustBrightness']

        for key in kwargs:
            if key not in available_funcs:
                print(f'{key} is not an available function. Exiting')
                break
            params = kwargs[key]
            function = getattr(self, key)
            if callable(function):
                output = function(self.result, params)
                self.result = output

    def flipHorizontal(self, matrix, params):...

    def flipVertical(self, matrix, params):...

    def rotate(self, matrix, params):...

    def perspectiveShift(self, matrix, params):...

    def cropResize(self, matrix, params):...

    def cropPaste(self, matrix, params):...

    def gaussianBlur(self, matrix, params):...

    def gaussianNoise(self, matrix, params):...

    def adjustSharpness(self, matrix, params):...

    def adjustBrightness(self, matrix, params):...
```

Methods – The Model

The model built had a 2D convolution layer followed by 2 linear layers.
The ReLu activation function was used between each layer

```
def forward(self, x):
    x = self.conv1(x)
    x = F.relu(x)
    x = x.flatten(start_dim=1)

    x = self.linear1(x)
    x = F.relu(x)

    logits = self.linear2(x)
    return logits
```

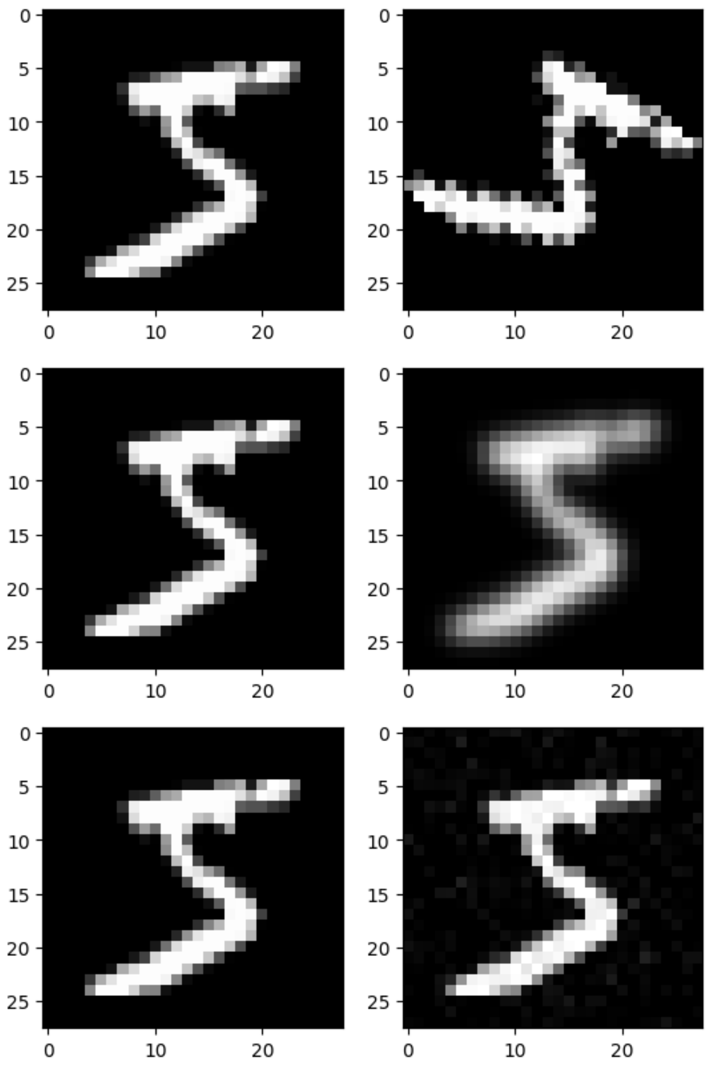
The model built differs from the one described in the method...

- Use case
- Computational resources

Methods – Training

The following compositions of augmentations were used to train the model on 1000 MNIST samples:

Geometric 1	Vertical flip, horizontal flip, rotate
Geometric 2	Perspective shift, crop+resize, crop+paste
Pixel 1	Gaussian noise, adjust brightness, Gaussian blur
Pixel 2	Adjust sharpness, adjust brightness, Gaussian Noise
Mixed 1	Vertical flip, horizontal flip, Gaussian Noise, crop+resize
Mixed 2	Rotate, perspective shift, adjust sharpness, adjust brightness
Mixed 3	Gaussian blur, rotate, crop+resize

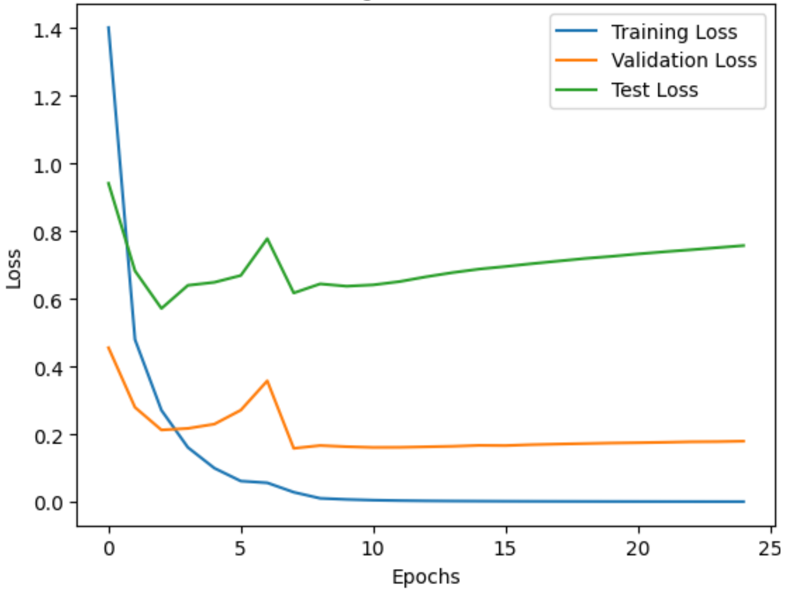


A random 90/10 training/validation split was used for the training dataset. Training was run for 25 epochs.

After each epoch, the model was evaluated on a 1000 sample test dataset.

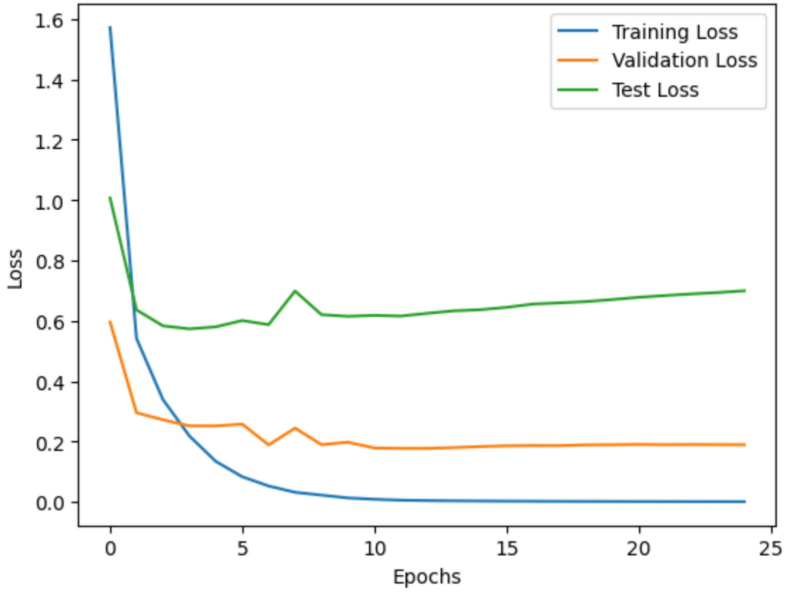
Results – Model Performance

Unaugmented



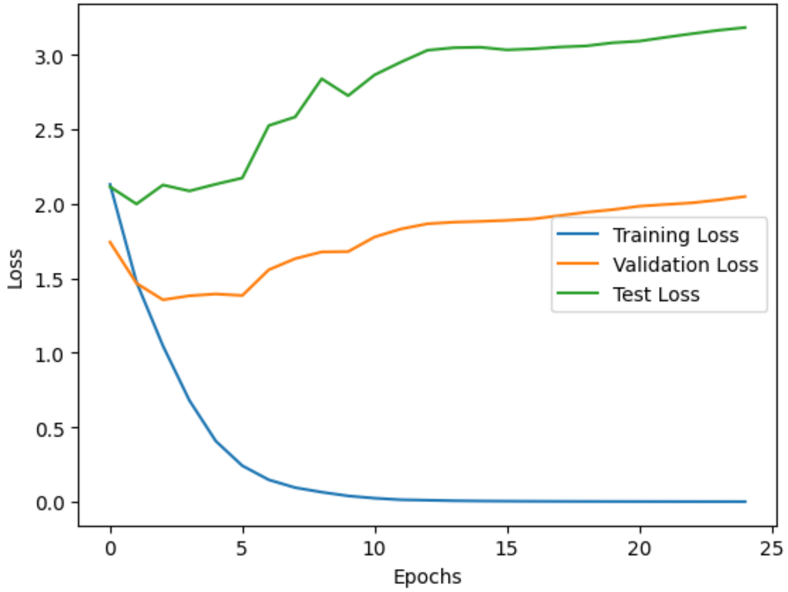
Min test loss: 0.58 @ epoch 3

Geometric 1



Min test loss: 0.59 @ epoch 3

Geometric 2

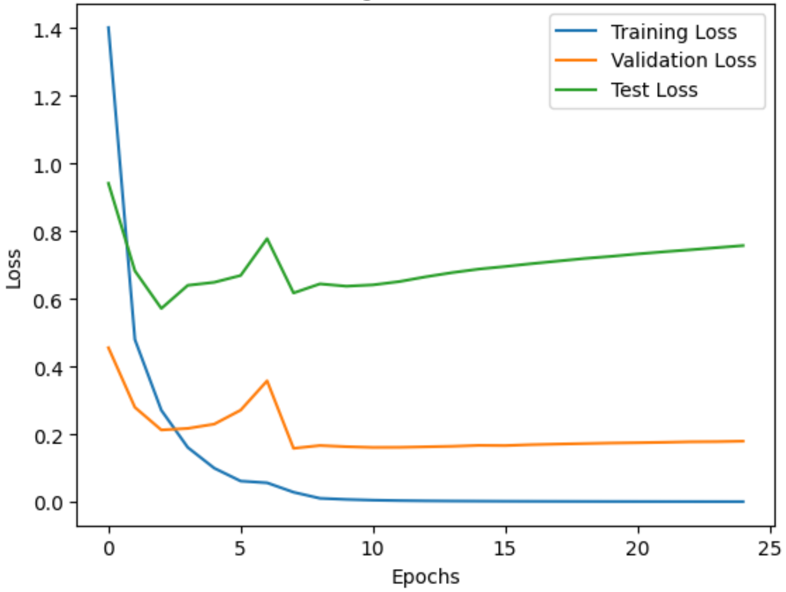


Min test loss: 2.03 @ epoch 2

Geometric 1	Vertical flip, horizontal flip, rotate
Geometric 2	Perspective shift, crop+resize, crop+paste

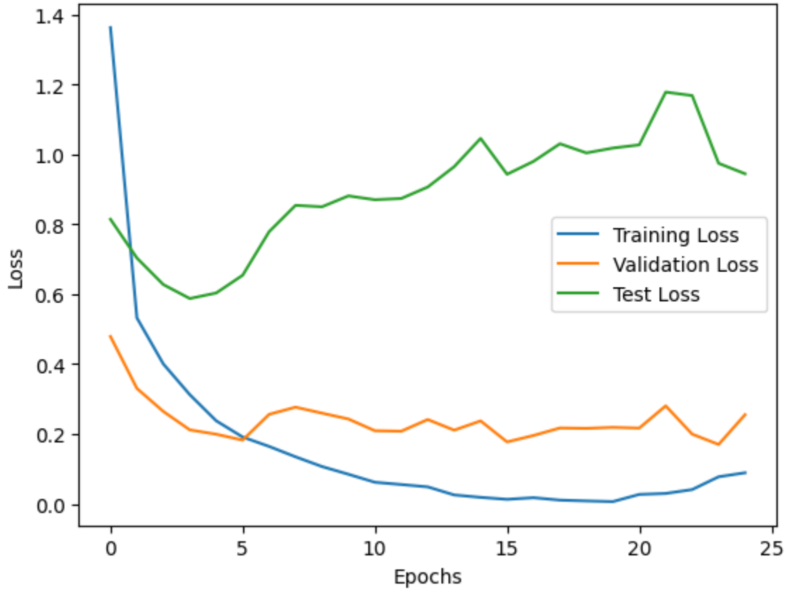
Results – Model Performance

Unaugmented



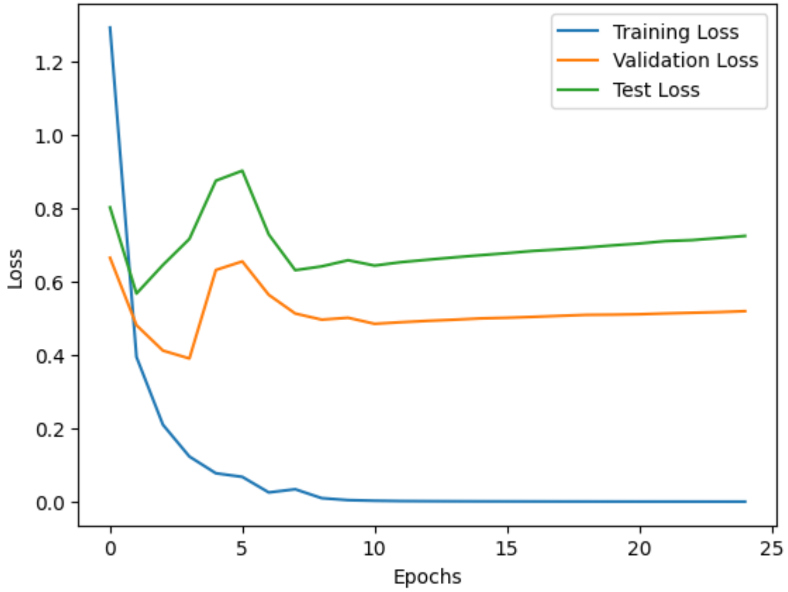
Min test loss: 0.58 @ epoch 3

Pixel 1



Min test loss: 0.59 @ epoch 3

Pixel 2

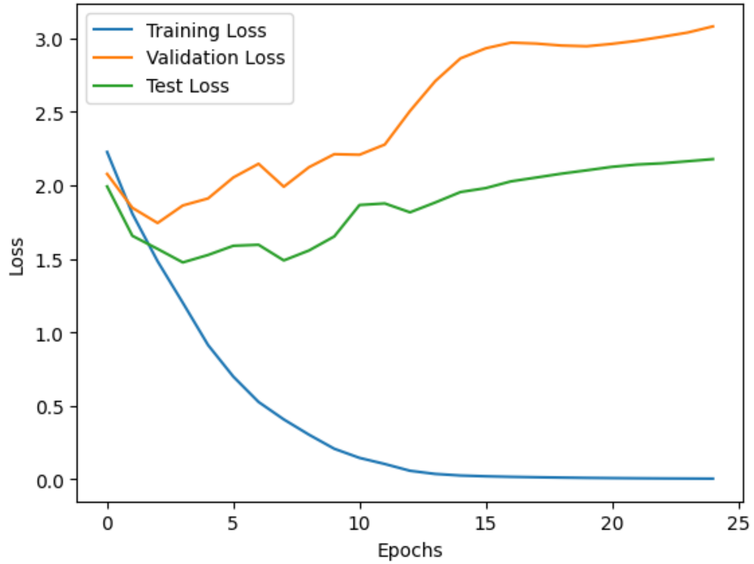


Min test loss: 0.58 @ epoch 2

Pixel 1	Gaussian noise, adjust brightness, Gaussian blur
Pixel 2	Adjust sharpness, adjust brightness, Gaussian Noise

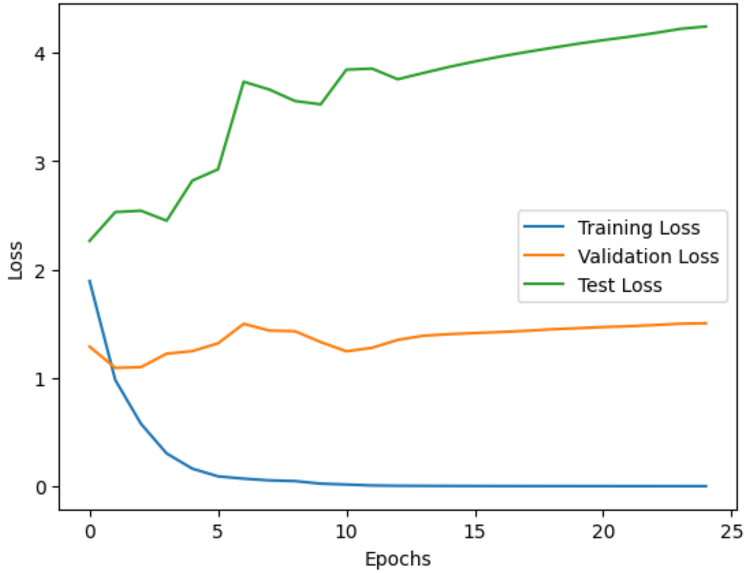
Results – Model Performance

Mixed 1



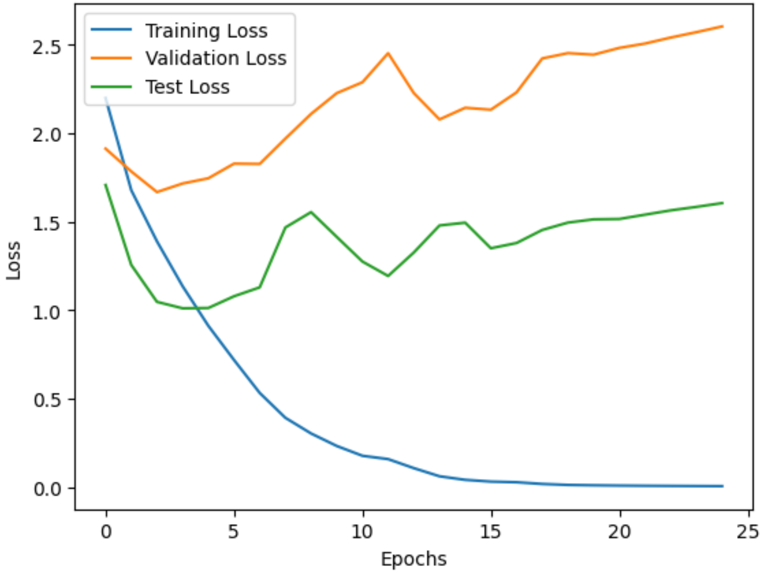
Min test loss: 1.50 @ epoch 3

Mixed 2



Min test loss: 2.31 @ epoch 1

Mixed 3



Min test loss: 1.03 @ epoch 4

Mixed 1	Vertical flip, horizontal flip, Gaussian Noise, crop+resize
Mixed 2	Rotate, perspective shift, adjust sharpness, adjust brightness
Mixed 3	Gaussian blur, rotate, crop+resize

Results – Challenges and Limitations

Aforementioned deviation from planned model architecture

Creating datasets is labor intensive

- augmentations applied beforehand a new images are saved

Limited number of augmentations

Discussion

The best performing augmentation compositions were the pixel-based methods and Geometric 1.

None of the compositions performed better than unaugmented training with only Pixel 2 equaling the unaugmented performance.

The mixed compositions performed the worst by a significant margin

Bibliography

1. Shorten, C., Khoshgoftaar, T.M. A survey on Image Data Augmentation for Deep Learning. J Big Data 6, 60 (2019). https://doi.org/10.1186/s40537-019-0197-0
2. Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18 (pp. 234-241). Springer International Publishing.
3. Deng, L. (2012). The mnist database of handwritten digit images for machine learning research. IEEE Signal Processing Magazine, 29(6), 141–142.
4. Kiran Maharana, Surajit Mondal, Bhushankumar Nemade, A review: Data pre-processing and data augmentation techniques, Global Transitions Proceedings, Volume 3, Issue 1, 2022, Pages 91-99, ISSN 2666-285X, https://doi.org/10.1016/j.gltp.2022.04.020.
5. A. Mikołajczyk and M. Grochowski, "Data augmentation for improving deep learning in image classification problem," 2018 International Interdisciplinary PhD Workshop (IIPhDW), Świnouście, Poland, 2018, pp. 117-122, doi: 10.1109/IIPHDW.2018.8388338. keywords: {Image color analysis;Machine learning;Lesions;Image classification;Neural networks;Cancer;Task analysis;Machine learning;style transfer;data augmentation;deep learning;medical imaging},
6. Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18 (pp. 234-241). Springer International Publishing.
7. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... Chintala, S. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. In Advances in Neural Information Processing Systems 32 (pp. 8024–8035). Curran Associates, Inc. Retrieved from http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf
8. Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, CJ Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E.A. Quintero, Charles R Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. (2020) SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. Nature Methods, 17(3), 261-272. DOI: 10.1038/s41592-019-0686-2.