What the f&nt?

Identifying typefaces through deep learning

Minna Kimura-Thollander, Katherine Sang, & Maggie Wu

Introduction

Fonts are the language of design, serving several industries, including advertising, publishing, and web development. In our project, we reimplemented a font recognition model known as DeepFont. Given an image of a font, the model can identify the font as well as find the top five most similar fonts to the given font.

Preprocessing

We used real and synthetic images from AdobeVFR. Due to the computational restrictions with GCP, we reduced the number of images and font classes that we trained on. Specifically, we eliminated typefaces in **bold** and *italics*.

Our Data Distribution

	Synthetic	Real
Autoencoder	150 000	1 920 000
DeepFont Train	750 000	0
DeepFont Test	150 000	60 000

Original vs. Our Implementation

	Ours	Original
Font classes	150	2 383
Image size	96 x 96	105 x 105
Total images	3 000 000	37 500 000

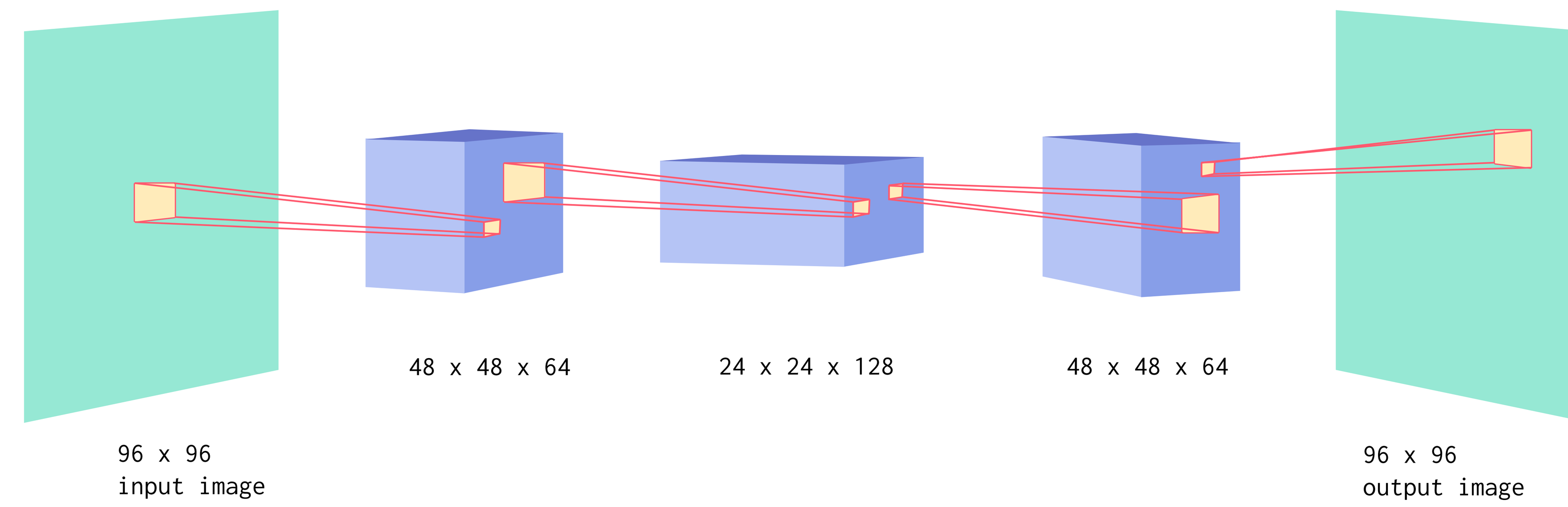
We applied noise, blur, perspective rotation, and shading to the images to prevent overfitting. Then we took 10 randomly cropped 96×96 px patches of each font image to train the model.

original image and random processed crops

Lastly, we shuffled the train and test inputs in groups of 10.

Model Architecture

Convolutional Autoencoder

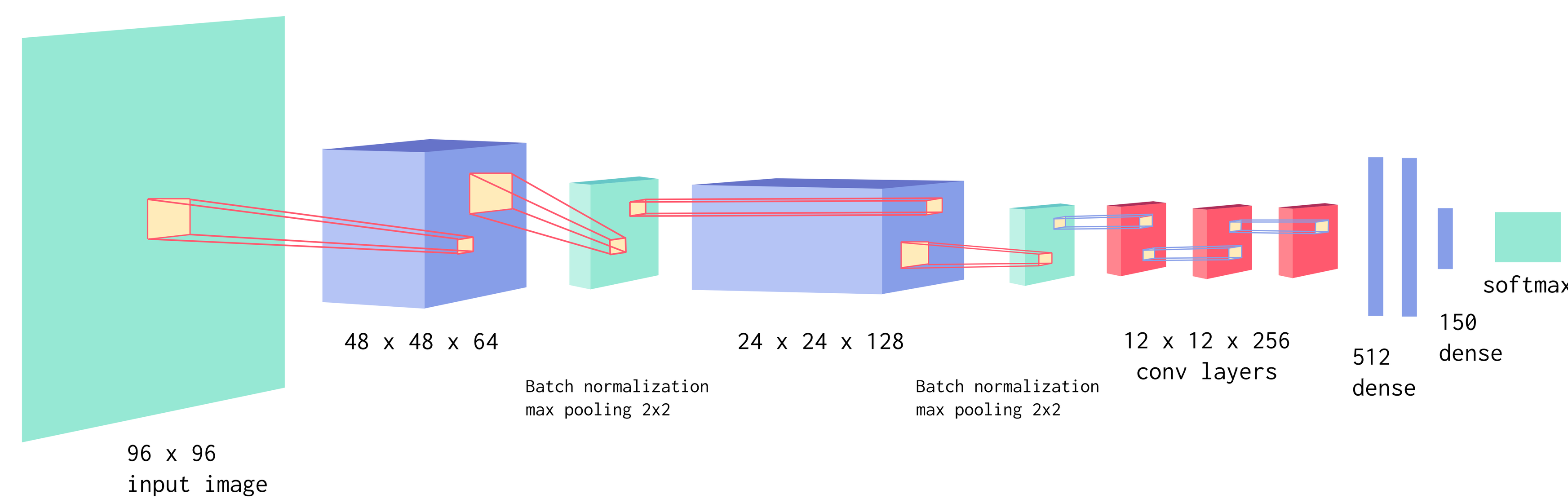


Hyperparameters

batch size	128
activation	leaky relu
learning rate	0.01(Adam)
epochs	5

This unsupervised network is used to learn lower level features of different fonts. The autoencoder compensates for the lack of labeled real world training data by exposing our model to a large number of unlabeled real world samples.

Deepfont Model (Original and Modified)



Hyperparameters

batch size	128
activation	leaky relu
learning rate	0.01 (Adam)
epochs	10

The DeepFont model is a supervised CNN. The first two convolutional layers are imported from the trained autoencoder, followed by three convolutional layers. We then use a series of dense layers, resulting in a probability distribution over our font classes.

We experimented with a modified version by increasing the filters on the last three convolutional layers and averaging across the *x* and *y* dimensions to produce a single value for each font class.

Challenges

Needing to handle a huge amount of data (~37.5 million images)

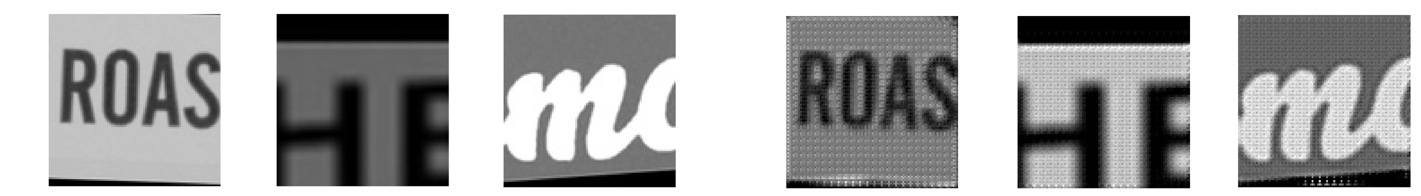
- Had to restructure model and data/labels for a smaller subset of images and fonts
- Not enough memory/disk space to process and store images/arrays
- BCF file unpacking required reverse engineering of the BCF packaging script

Poor documentation and unclear implementation of original model

- No information on kernel sizes, epoch count, strides, or activations
- Lack of clarity with how the autoencoder layers were used in the DeepFont model
- Unclear explanation of how preprocessed images enter the model

Results

Convolutional Autoencoder



Original inputs   Autoencoder outputs

Loss (calculated w/ MSE)

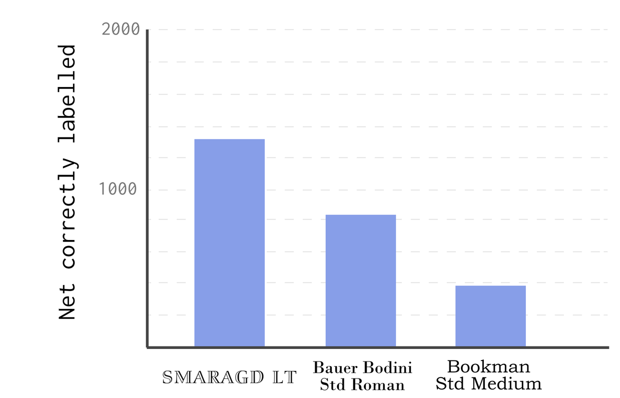
Epoch	1	2	3	4	5
Loss	26.610722	0.477306	0.2397055	0.070799	0.0067023

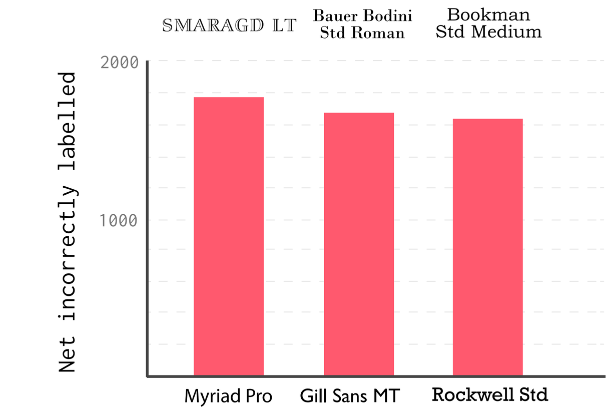
Deepfont Model

Loss (calculated w/ sparse categorical cross entropy)

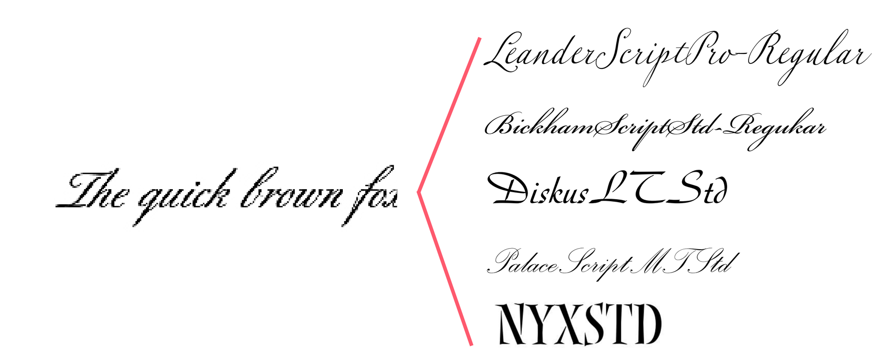
Epoch	2	4	6	8	10
Original	2.1808722	0.386900	0.1625434	0.1308935	0.1159054
Modified	0.4263132	0.127855	0.0828373	0.067483	0.0602210

Best and Worst Performing Fonts





Font Identification Tests



Discussion & Future Work

Lingering issues

- Accuracy is abnormally high starting from first epoch (89%)
- Low performance on sans serif fonts and our own real world samples despite high accuracy
- Model unable to identify more than one font at a time in a single image

Improving data & methodology

- Updating the AdobeVFR dataset to include ubiquitous web fonts (Lato, Proxima Nova, Roboto) and data sourced from digital media, not just print and physical
- Obtaining more computational resources to process the entire AdobeVFR dataset
- Tweaking hyperparameters for autoencoder and DeepFont model

Sources

Original paper: "DeepFont: Identify Your Font from an Image" by Zhangyang Wang, Jianchao Yang, Hailin Jin, Eli Shechtman, Aseem Agarwala, Jonathan Brandt, Thomas S. Huang.

Special thanks to Professor Ritchie and Mounika Dandu. ❤