Individual Character Handwriting Reading

Team Tangerine: Michael Aiello & John Rosso

Problem Statement:

Amidst different, or perhaps aesthetically similar writing styles that each person might have, the main goal and problem to be solved is how to read these different handwriting styles using deep learning methods. Given the time we have in a semester and with the level of expertise we have relative to the more difficult, full complexity of the larger problem in reading and understanding full words and sentences, our project aims to just solve the subsequent problem of correctly reading and deciphering individual handwritten characters first, the building blocks of more full words and sentences.

Proposed Approach:

For this project, we propose to create a working model that utilizes deep learning methods like convolutional neural networks to determine which handwritten English character is being shown to the model. To do this we will use a dataset that contains pictures of handwritten English characters, and using this dataset we will design and train a working deep-learning model. This model will be changed and experimented on by using different formats and set up of layers to help achieve a maximum performance metric. Once the best model we can find for our dataset is achieved, a final test set metric will be produced and recorded. We can then potentially utilize the best model to solve the subsequent problem in the statement above in order to work towards the main goal.

Real-World Application:

Professional, advanced, Handwriting reading software and tools (OCR - Optical Character Recognition, etc) already exist in many real-world applications. There are scanners out there that are used across all different industries to digitize anything from printed, handwritten documents, to forms, questionnaires, and submission documents. In education for example many standardized tests are either evaluated or consolidated/organized with these technologies.

In a realist sense, the tangibility of using our model in a real world application is probably low considering the juxtaposition of our small scale school project to a real world technology funded and developed by large corporations and entities. It's also important to note that our model is for individual English characters, not full words and sentences, and it's this distinction that unfortunately separates itself to probably not be applied in more professional, advanced contexts.

Despite this, idealistically, if our model can be used as a baseline for another model, it would serve as building blocks for developing a more advanced handwriting reader that can understand the full words and sentences as previously mentioned above. If anything, the model could contribute or serve as the baseline level of work needed for developing a more advanced

automated student grading technology to support SCSU school operations. The implications along with the debate on the necessity or potential of implementing technologies of this nature would require a more indepth, professional, holistic examination and approach in order to be considered for a legitimate real world application endeavor.

Dataset:

- **Description:** 3,410 images of handwritten English characters (0-9, a-z, A-Z) created by de Campos, T.~E. and Babu, B.~R. and Varma, M. This dataset is sourced from Kaggle at:
  https://www.kaggle.com/datasets/dhruvildave/english-handwritten-characters-dataset?resource=download
  and is licensed under the Open Data Commons Open Database License.

- **Statistics:** Each class contains exactly 55 images. Each image is originally sized at 1200 * 900 pixels in PNG formats.

- **Preprocessing:** Normalization of batches and random selection are used in creating set splits. Images are resized to 224 * 224 pixels.

- **Sample images:** (before / After)

Zero: **One:**

Two: **Three:**

Four: **Five:**

Six:

Seven:

Eight:

Nine:

A:

B:

C:

D:

E:

F:

G:

H:

I:

J:

K:

L:

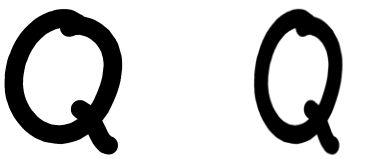
M:

N:

O:

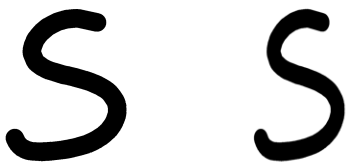
P:

Q:



R:

S:



T:

U:

V:

W:

X:

Y:

Z:

a:

b:

c:

d:

e:

f:

g:

h:

i:

j:

k:

l:

m:

n:

o:

p:

q:

r:

s:

t:

u:

v:

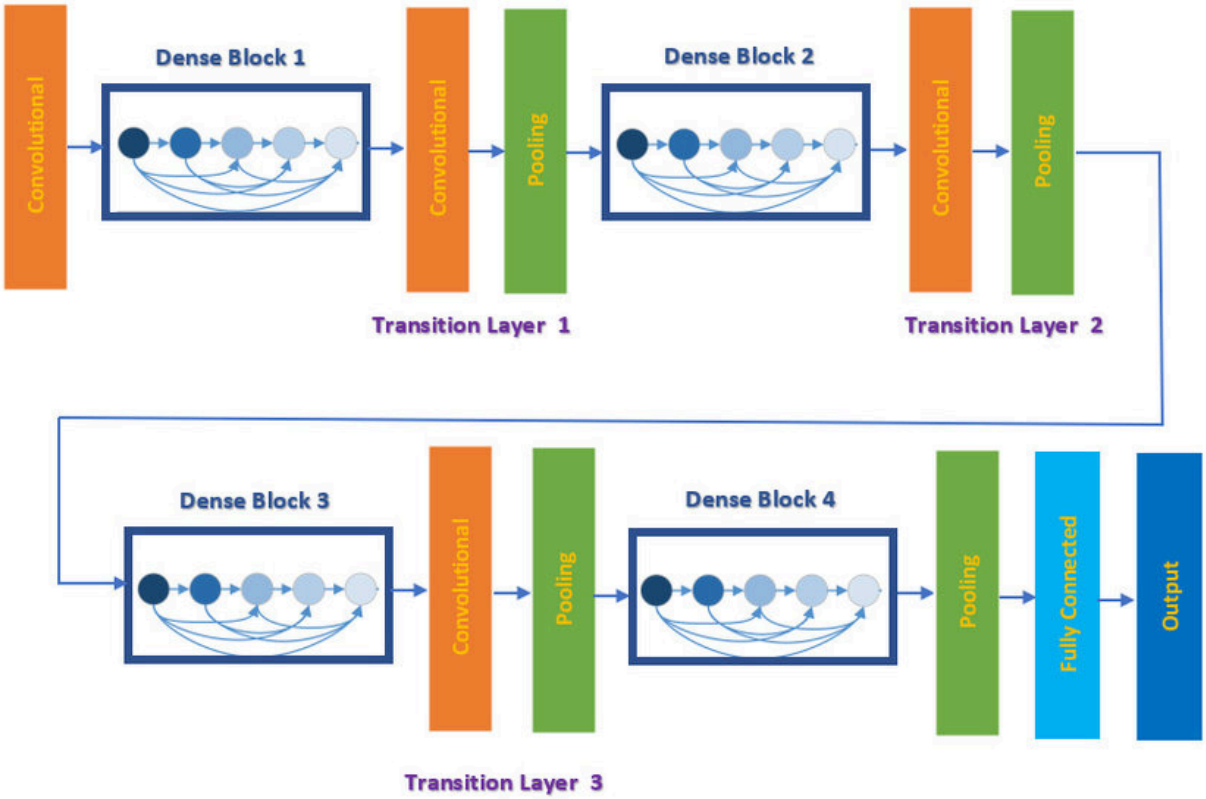
w:

x:

y: **z:**

Methods:

- **Proposed Model:**
    - DenseNet201: Series of dense blocks consisting of multiple convolutional layers. DenseNet uses outputs of previous blocks as an input to the current processed dense block. This network also concentrates output feature maps with incoming feature maps rather than summarizing them like ResNet does.



- **Candidate Models:**
    - VGG_19: A convolutional neural architecture that utilizes 13 layers with 3x3 filters stacked into five blocks, each followed by a max pooling layer for downsampling, or essentially making the image smaller to focus on the more important parts. The repetitive stack-like structure extracts the patterns in the images like edges, colors and textures.
    - ResNet152: Has a core building block called the residual block consisting of two pathways, the traditional convolutional layer and then a direct identity connection bypassing the convolutions. The output of the layers is added to the unchanged input with the identity connection. The network learns from the sum of original input and the transformed features, effectively preventing gradients from vanishing as they travel through the layers. These residual blocks are stacked in conjunction with other convolutional layers, batch normalization, and activation functions, summing to a total of 152 layers.

- **Baseline Model:**

Convolutional Neural Network model with random selection, normalization, convolutional layers, max-pooling layers, and fully connected layers. (Input > Conv/Activation > Pooling > Conv/Activation > Pooling > Fully Connected)

Experiment:

Hyperparamters	**VGG_19**	**DenseNet201**	**ResNet152**
Epochs	15	15	15
Learning Rate	0.00120226445142 17973	0.00144543976057 32083	0.00057543994626 03033
Early Stopping	Delta(.0001), Patience(5)	Delta(.0001), Patience(5)	Delta(.0001), Patience(5)

Results:

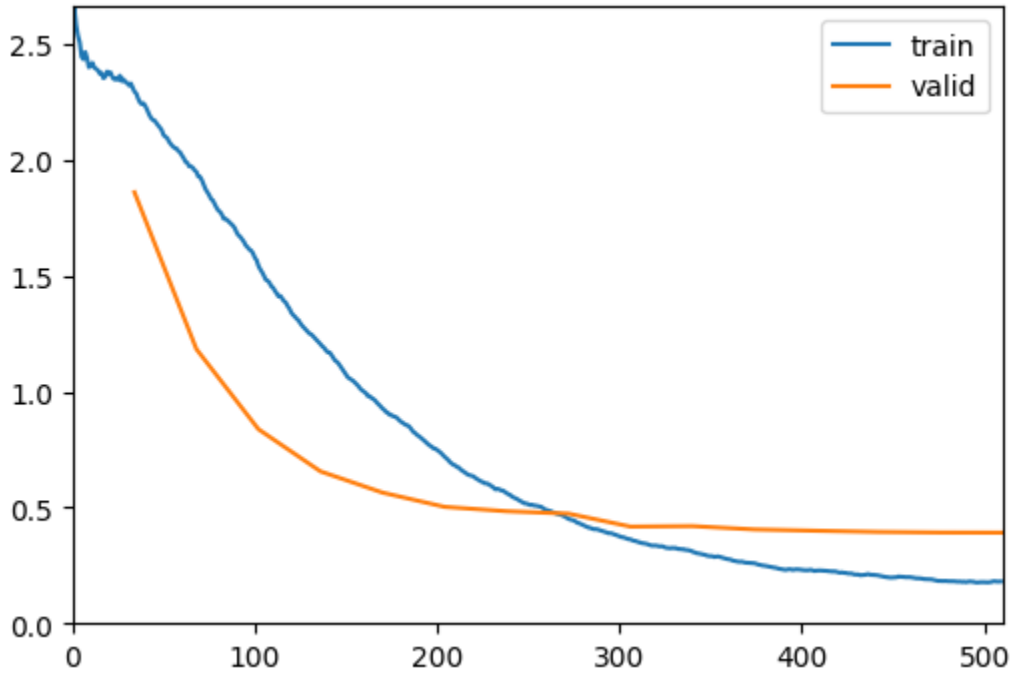
Validation Set:

Metrics	Baseline	VGG_19	ResNet152
Accuracy	.5614	0.856881	0.866055
Precision	.5639	0.851062	0.856599
Recall	.5142	0.857707	0.860478
F-1	.5087	0.847365	0.850707

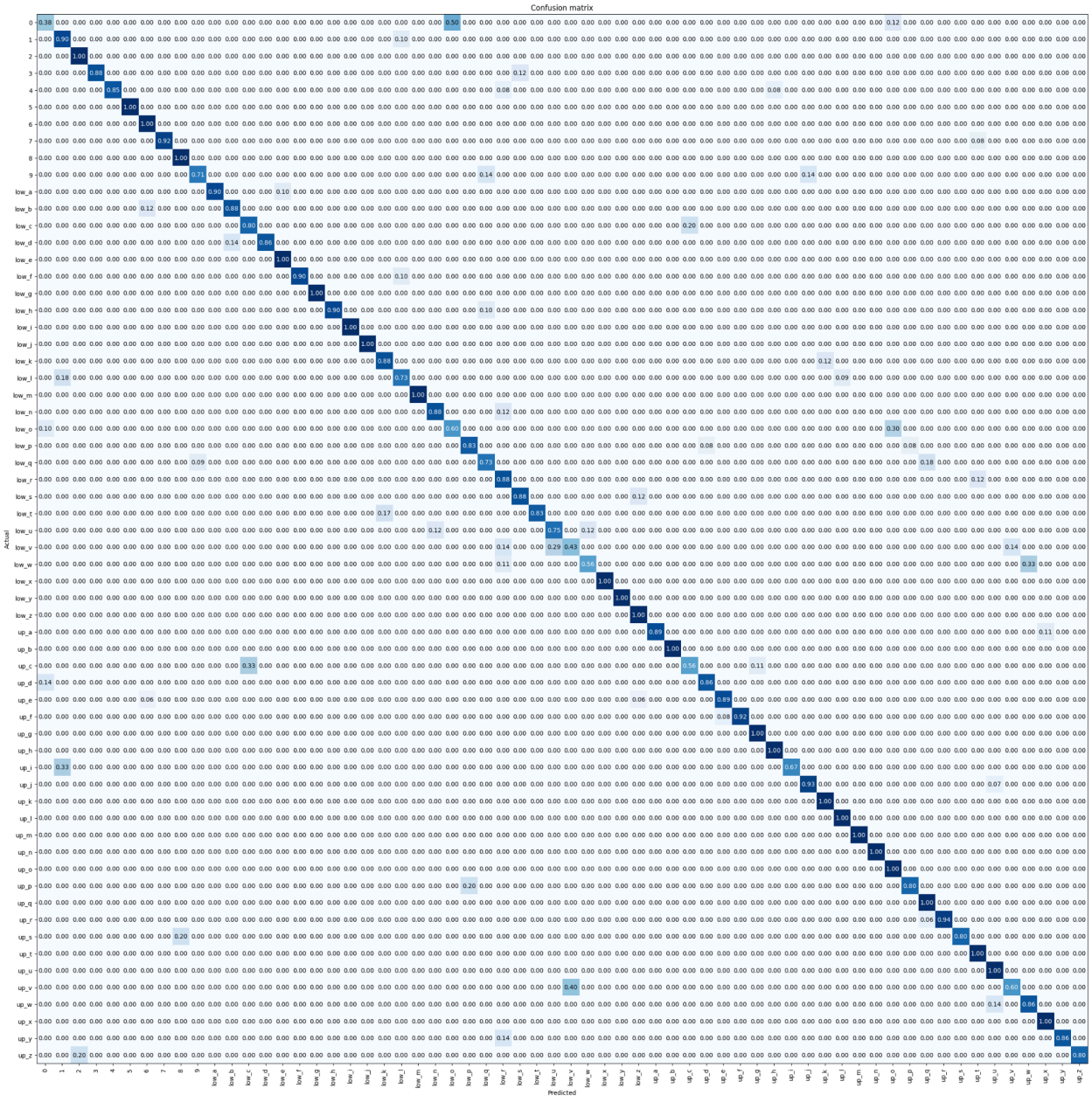
Test Set:

Metrics	VGG_19	DenseNet201	ResNet152
Accuracy	0.885630	0.868035	0.848974
Precision	0.885351	0.876785	0.853380
Recall	0.884917	0.865739	0.851173
F-1	0.880034	0.865309	0.843673

Loss Curve: DenseNet 201 Test Set



Confusion Matrix: DenseNet201 Test Set



Discussion:

- **Challenges:**
    - Hardware limitations when experimenting with larger networks (out of memory)
    - Trying to get similar results when switching to different hardware. Often found better results from one set of hardware compared to another.
    - Keeping track of which code cells to run and which to not. This often ruined experiments and the data needed to be reloaded into memory.

- **Insight:**
    - Using “Cuda Cores” made experiments take a fraction of the time they would on a CPU.
    - Difficult to achieve high accuracy results (>90) even when using pre training.
    - Using Python notebooks helped keep experiments organized and readable.

- **Improvements:**
    - More image augmentations.
    - Better dataset with more images and distinct writing styles
    - More fine tuning.

Conclusion:

Though an accuracy score of over 90% was not achieved there were scores of high 80’s; with a max accuracy score of 88% on the validation set. This gave good insight as to the difficulties in achieving high accuracy using deep learning with what seems to be an easy classification task. With better image augmentation and potentially a larger dataset maybe high accuracy can be achieved. It was also noticed how much computational power is needed to accomplish training these network models. It has definitely been an eye opening experience to the ever changing world that is deep learning.