UE17CS490B – Capstone Project Phase – 2

SEMESTER - VIII

END SEMESTER ASSESSMENT

Project Title	: Style Consistent Kannada Font Generation
Project ID	: PW21KS04
Project Guide	: Prof. K S Srinivas
Project Team	: Saahil B Jain - PES1201700241
	Jeevana R Hegde - PES1201700633

Outline

Abstract

Problem Statement

The project is to automate the generation of new fonts for the Kannada scripts while taking inspiration from existing fonts from English language.

Abstract

Introduction

- **Indian languages** currently have a **limited** set of **fonts** available to choose from.
- **Creation** or generation of new fonts is at **present**, a **manual** process.
- This can tend to be a **tedious, expensive and time-consuming** process.

Abstract

Introduction and objective

- Our project **goal initially** was mainly to **automate** the **generation** of **new fonts** for the **Kannada** scripts while **simultaneously** ensuring **generation** of **ಮಾತ್ರಗಳು (maatras) and ಒತ್ತಕ್ಷರಗಳು (otaksharas).**

Abstract

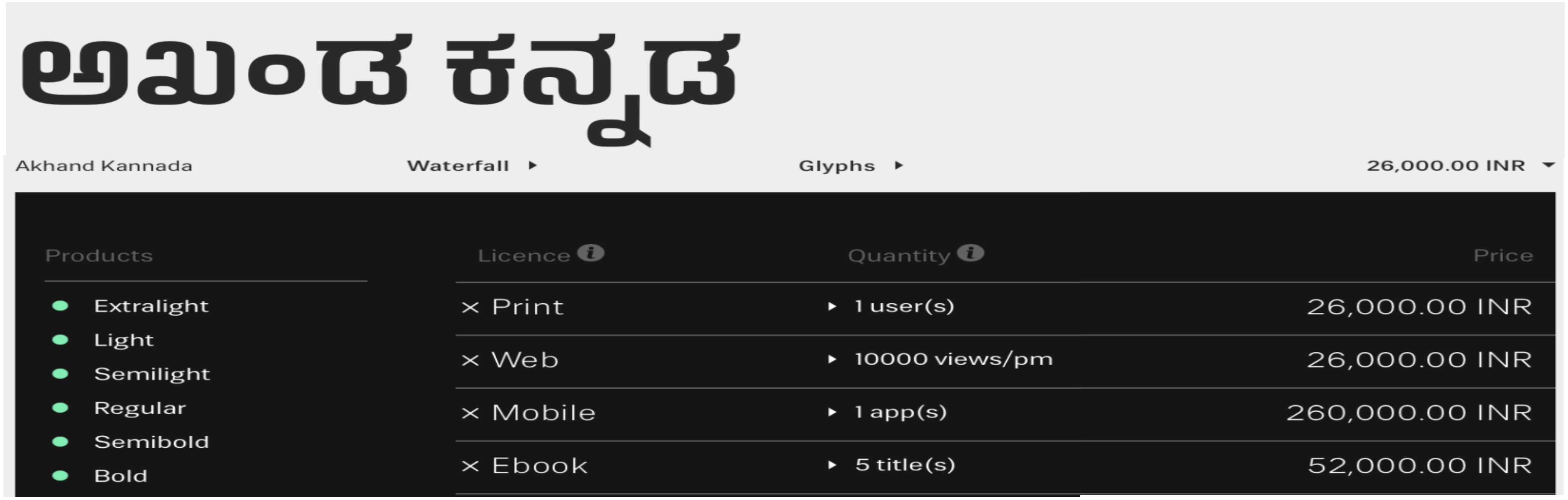
Introduction and objective

- Our current objective leans towards **taking inspiration** from **existing fonts from** a language such an **English** which has an abundance of styles and **inculcating** these in creating **new fonts for Kannada.**
- This is essentially **Style-Transfer**.

Abstract

Scope

- Indian Type Foundry (ITF) is a company which **manually** creates customized fonts for various languages.
- The charges range from **26,000INR - 2,60,000INR** for a specific typeface.



ಅಖಂಡ ಕನ್ನಡ

Akhand Kannada | Waterfall ▸ | Glyphs ▸ | 26,000.00 INR ▾

Products	Licence	Quantity	Price
● Extralight	× Print	▸ 1 user(s)	26,000.00 INR
● Light	× Web	▸ 10000 views/pm	26,000.00 INR
● Semilight	× Mobile	▸ 1 app(s)	260,000.00 INR
● Regular	× Ebook	▸ 5 title(s)	52,000.00 INR
● Semibold			
● Bold			

Abstract

Scope

- Currently font generation is done where the designers must **manually draw** and further **trace Glyphs** (characters) using software tools such as **Font Developer**.
- For each font, the time varies from a couple of hours to a couple of days.
- Our project aims to **automate** this process which makes the process **less time-consuming**, **more efficient(using style consistency)** and provides a **varied set of options** to choose from at a **cheaper price**.

Abstract

Adopted approach

- For the **generation** of the **fonts**, we went ahead with **Neural Cellular Automata** approach. Here, the update rule is learnt using its neighbours (for each cell) and this is repeated multiple times to generate a new font.
- **Style transfer** is done using **Neural Style Transfer** where this incorporates the style from whichever English font's character is provided as input as provides consistently styled output for the Kannada character.

Team Roles and Responsibilities

Phase	Student Name	Contribution
Literature survey	Jeevana	Typeface Completion with GANs (Research paper)
	Saahil	Attribute to Font (Research Paper) , Multi-Content GANs (Research paper)
Data collection	Jeevana	Wrote script to collect glyphs of some English and Kannada fonts using python
	Saahil	Wrote script to crop glyphs to enhance image using python
	Both	Collected and cleaned a few fonts for English and Kannada
Testing new methods	Jeevana	Testing Neural Style Transfer - Generated Locomo fonts for Kannada
	Saahil	Testing Autoencoders and Multi-Content GANs: Dropped due to limitations
Generating Fonts	Jeevana	Generated Picket fonts for Kannada
	Saahil	Generated Schmalfette fonts for Kannada
Cleaning fonts	Both	Cleaned the fonts generated using script

Summary of Requirements

Summary of Requirements

- Basic Glyph of all letters with all maatras and ottaksharas of the Kannada Script.
- Glyph of at least one complex letter (“g”, “b”, “R”) of target font of English.
- A GPU to generate fonts, as time to generate on CPU would be extremely large. Google colab/ Kaggle notebooks would provide sufficient GPU power.

Strengths and Weaknesses

State of the art review:
As per the current manual method of doing this,

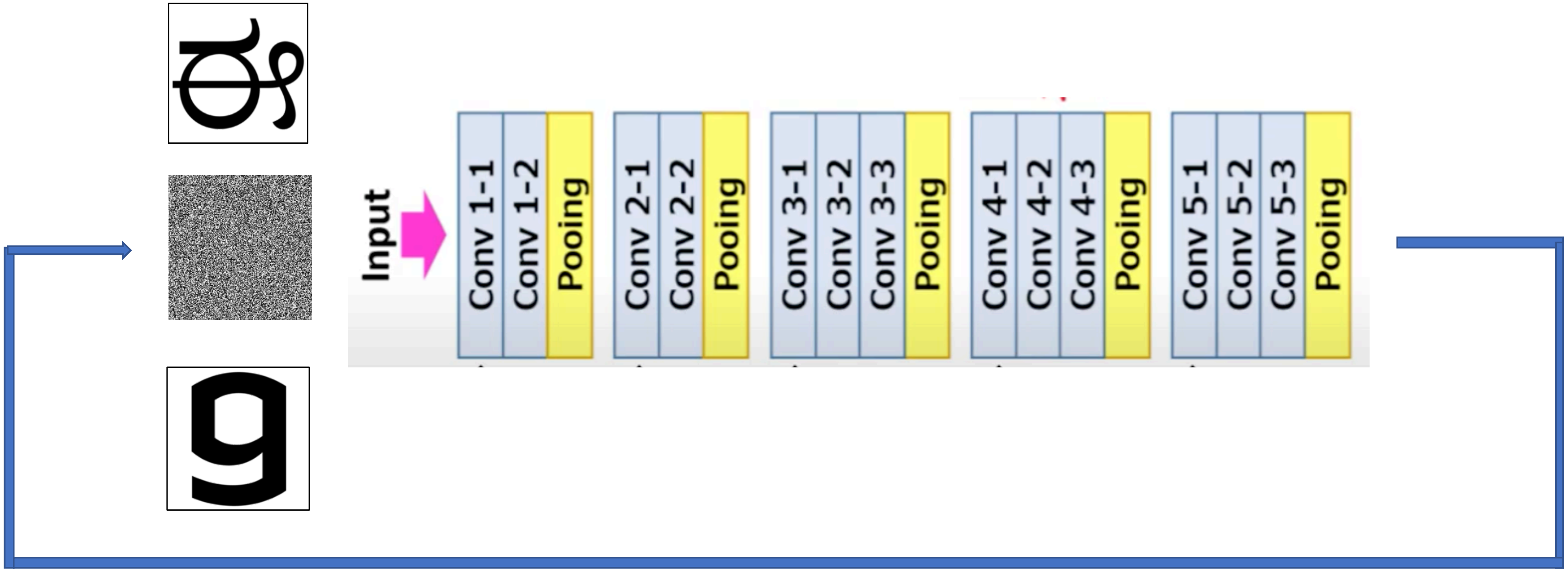
Strengths:
1. Customized to user's expectations
2. In case of specific word, there is not need to generate fonts for all characters.

Weaknesses:
1. Time-consuming
2. Expensive
3. Lesser variety

Summary of Approach

Proposed approach: Neural style transfer

Model Architecture:



Summary of Approach

Details of the approach:

- The **inputs** include the **style image, target character and the generated image**.
- The three inputs are passed to the Neural network **iteratively** and the **loss calculated is back propagated**.
- The **output** of **each iteration** is passed as **input** to the **next iteration**.
- The process is **repeated** until the **generated image is satisfactory**.

Summary of Approach

Benefits:

1. Automated
2. Fast
3. Can take inspiration from not just fonts but also from a variety of sources.
4. Many options to choose from with different style inputs.

Drawbacks:

1. The fonts generated might need little cleaning.
2. GPU is required.
3. Each letter is generated individually, so learning from one letter can’t be used to generate a second letter.

Design Description

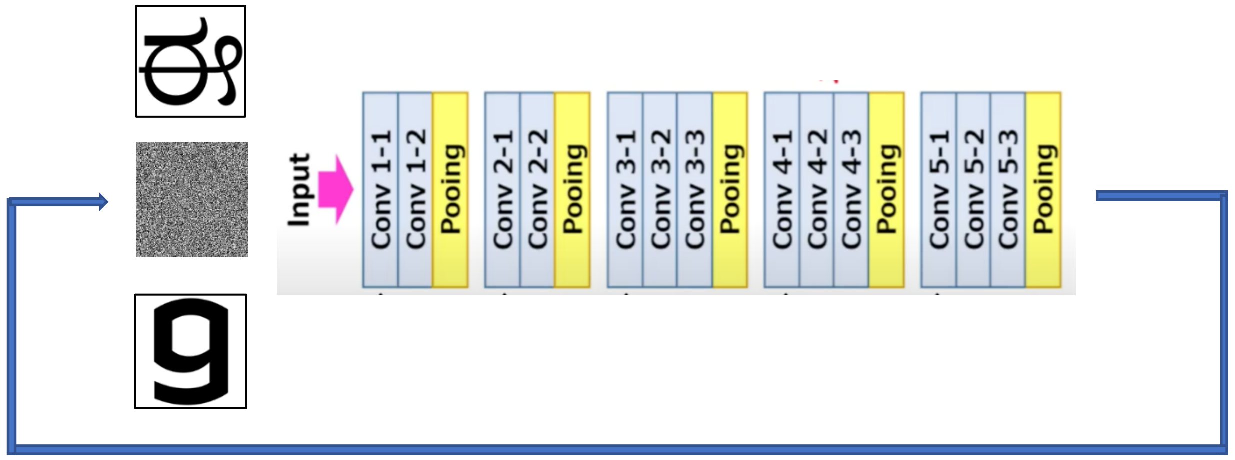
We Initially tried out **multiple methods** including **GANs** and **Convolutional Autoencoders**.
However, these methods had their own **limitations**:

1. **GANs** were **dependent** on a **large amount of data** being available, which is **not available** for Indian languages such as **Kannada**.
2. **Convolutional Autoencoders** were able to achieve style transfer to some degree, however the **performance plateaued after** a **certain depth** of the model and the results weren't satisfactory.

Hence, the **algorithm** we use is - **Neural style transfer.**

Design Description

Neural style transfer is an **iterative** optimisation technique that takes **three** images as **input**:



1. **Target image-** which is the character of the Kannada script we want to convert to the desired font.
2. **Content image-** which is the actual generated image which is passed iteratively through the network.
3. **Style image-** which is an alphabet of English in the desired font style, in this case ‘g’ in Locomo style .

Design Description

For **every iteration, a loss** is **calculated** and **back propagated** through the network.

These **three input** images are **passed iteratively** through the **network till** the result, that is the **content image is satisfactory**.

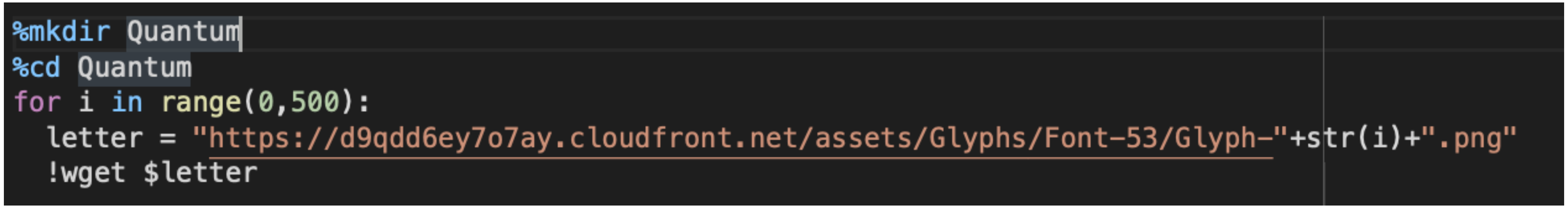
Modules and Implementation Details

Module name: Data collection

Technology used: Wget which is a computer program that retrieves content from web servers.

Description:

Collecting data includes **collecting multiple fonts of English language and a few available fonts of Kannada language**. Our code fetches all the characters from ITF where we only need to provide the link to the glyph page of the font we need.



```
%mkdir Quantum
%cd Quantum
for i in range(0,500):
  letter = "https://d9qdd6ey7o7ay.cloudfront.net/assets/Glyphs/Font-53/Glyph-"+str(i)+".png"
  !wget $letter
```

Modules and Implementation Details

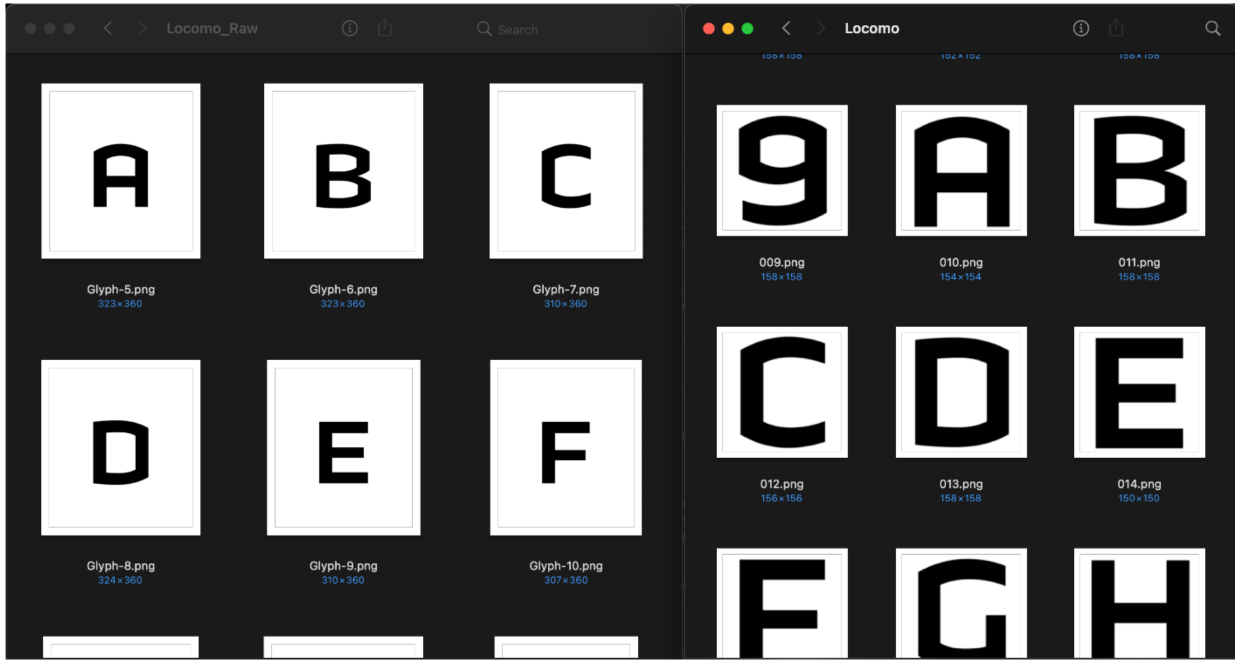
Module name: Data cleaning

Technology used: Python

Description:

Once, these glyphs are downloaded, we **get rid of the additional padding** for the images to be visible clearly to our model. We also ensure **images are centered** and have an **aspect ratio of 1:1** (meaning they are squares).

Before **After**



Modules and Implementation Details

Module name: Generating fonts

Technology used: Python

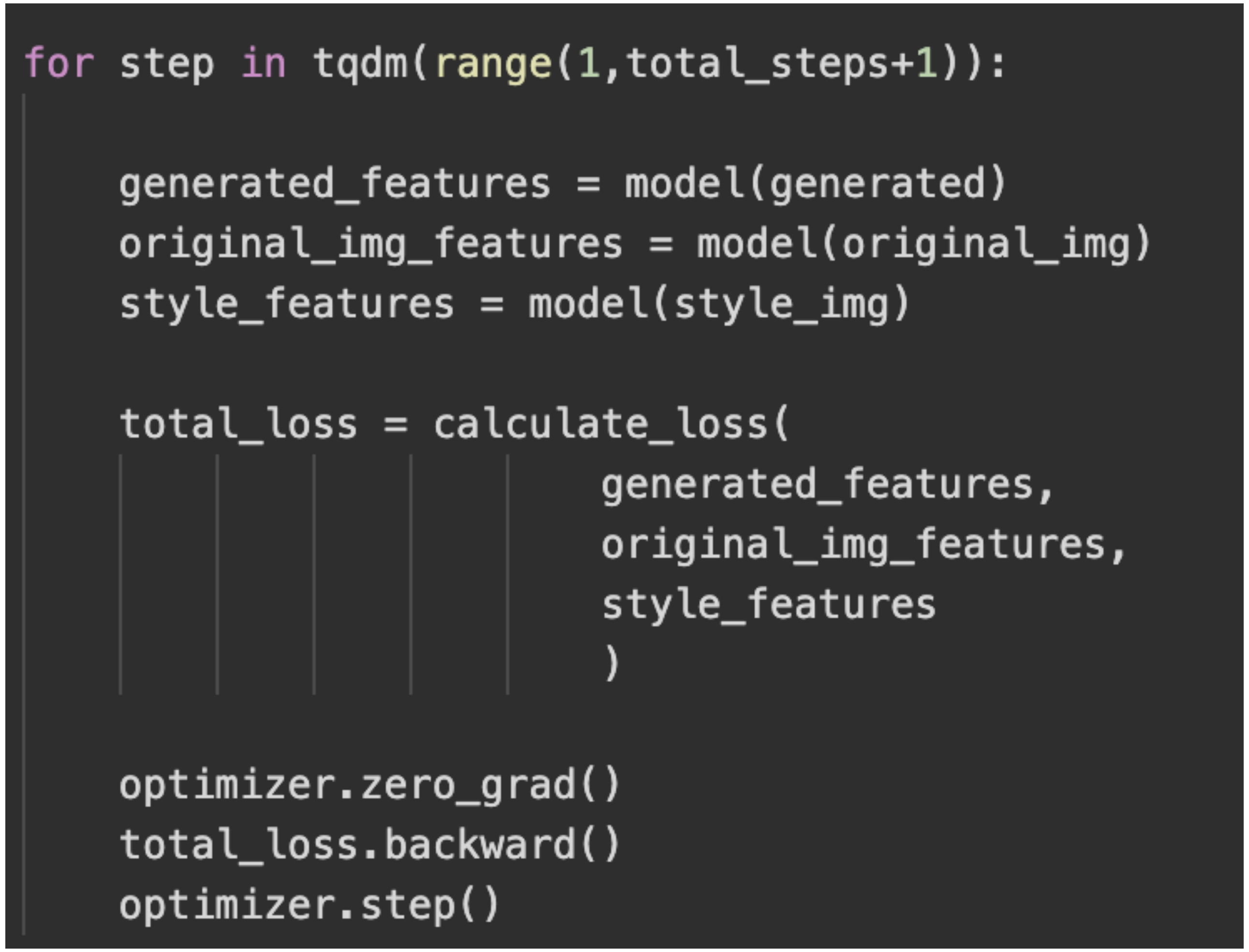
Description:

The most important step of our project is actually **generating the fonts**. We need to **iterate** over the set **epochs** each time, we provide the **three images** and **input** to the network.

Calculate the **loss** over the output and finally backpropagate this loss.

Modules and Implementation Details

Generation of fonts:



```
for step in tqdm(range(1,total_steps+1)):

    generated_features = model(generated)
    original_img_features = model(original_img)
    style_features = model(style_img)

    total_loss = calculate_loss(
                                generated_features,
                                original_img_features,
                                style_features
                                )

    optimizer.zero_grad()
    total_loss.backward()
    optimizer.step()
```

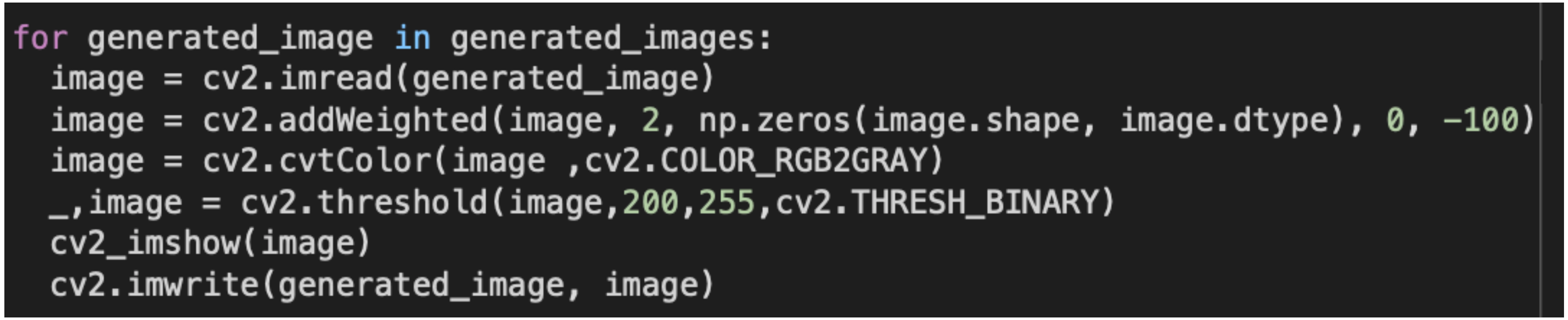
Modules and Implementation Details

Module name: Cleaning generated fonts

Technology used: Python

Description:

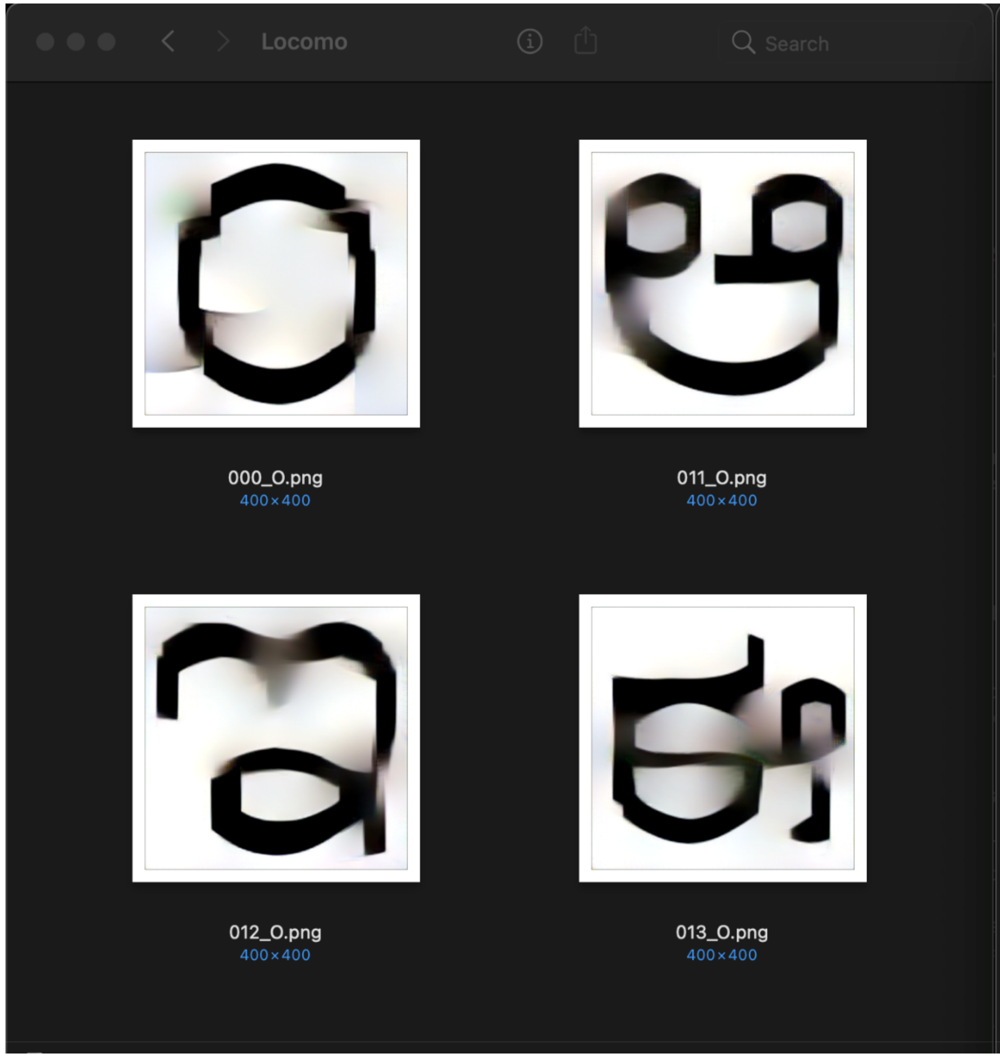
The generated fonts are cleaned and refined to obtain expected results.



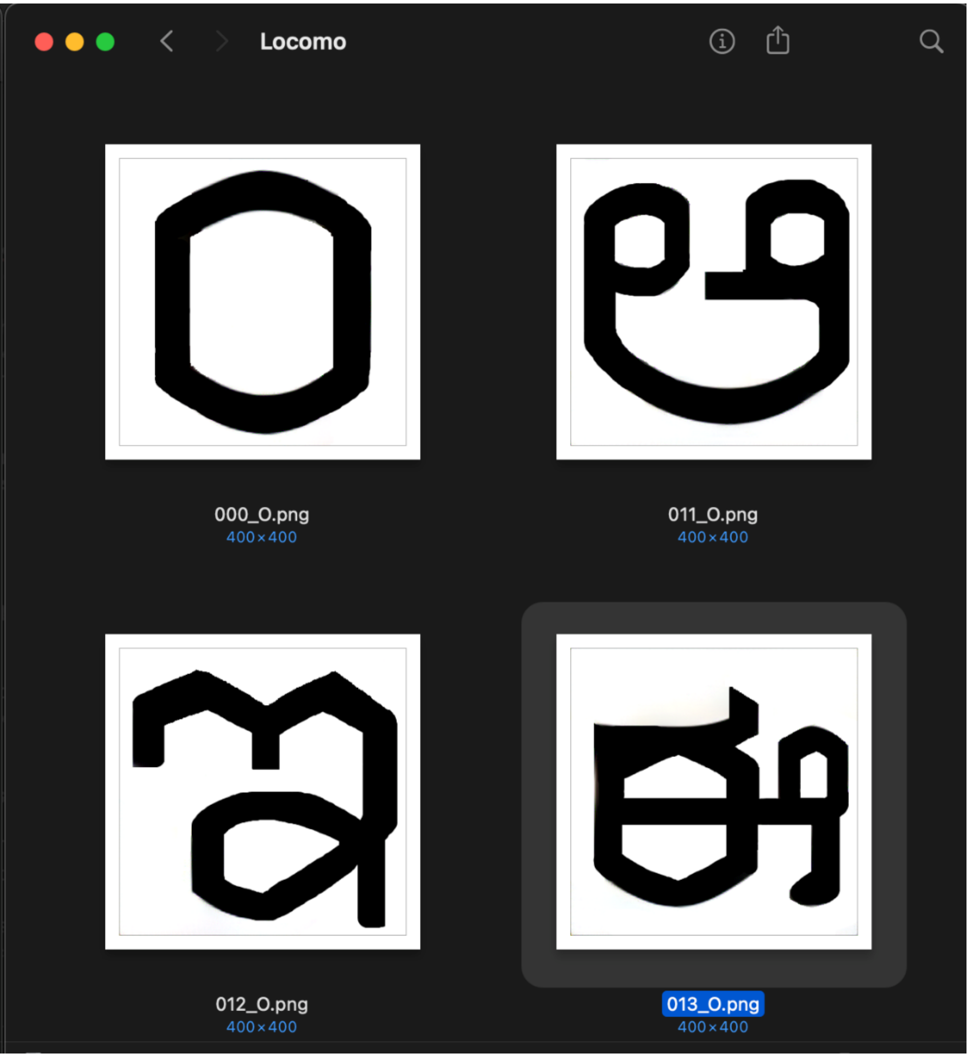
```
for generated_image in generated_images:
  image = cv2.imread(generated_image)
  image = cv2.addWeighted(image, 2, np.zeros(image.shape, image.dtype), 0, -100)
  image = cv2.cvtColor(image ,cv2.COLOR_RGB2GRAY)
  _,image = cv2.threshold(image,200,255,cv2.THRESH_BINARY)
  cv2_imshow(image)
  cv2.imwrite(generated_image, image)
```

Modules and Implementation Details

Before



After



Project Demonstration

Demonstration walk through

Test Plan and Strategy

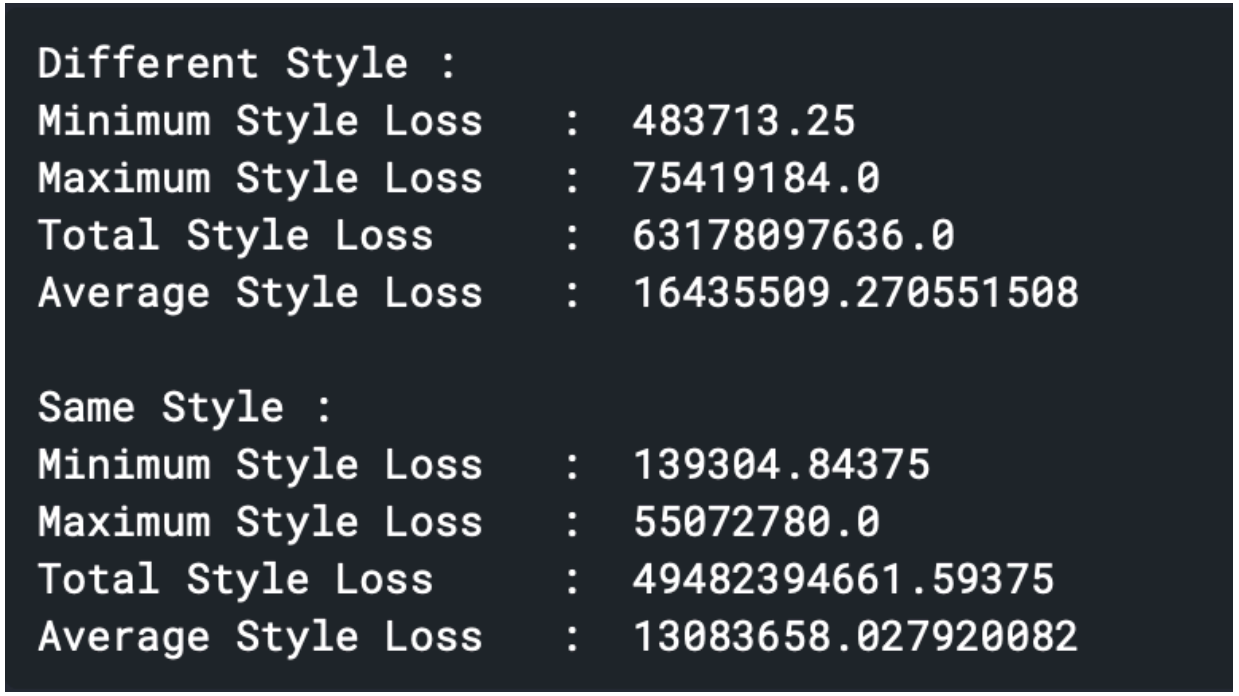
Non-functional performance test:

- We wrote a script to help us **evaluate** the **loss function**.
- This script took **two font styles** as **input** and **took all combinations of letters pairs of the first font and second font as well as all combination of letter pairs of the second font with itself**. **Loss** was calculated for **each pair** and the results were compared.
- **Ideally** the **style losses** for **combinations** of letters pairs of the **first font and second font** should be **really high** and **style loss** of letter pairs of the **second font with itself** should be **minimal**.

Test Plan and Strategy

Choosing the right Loss Function:

Default Style Loss Function

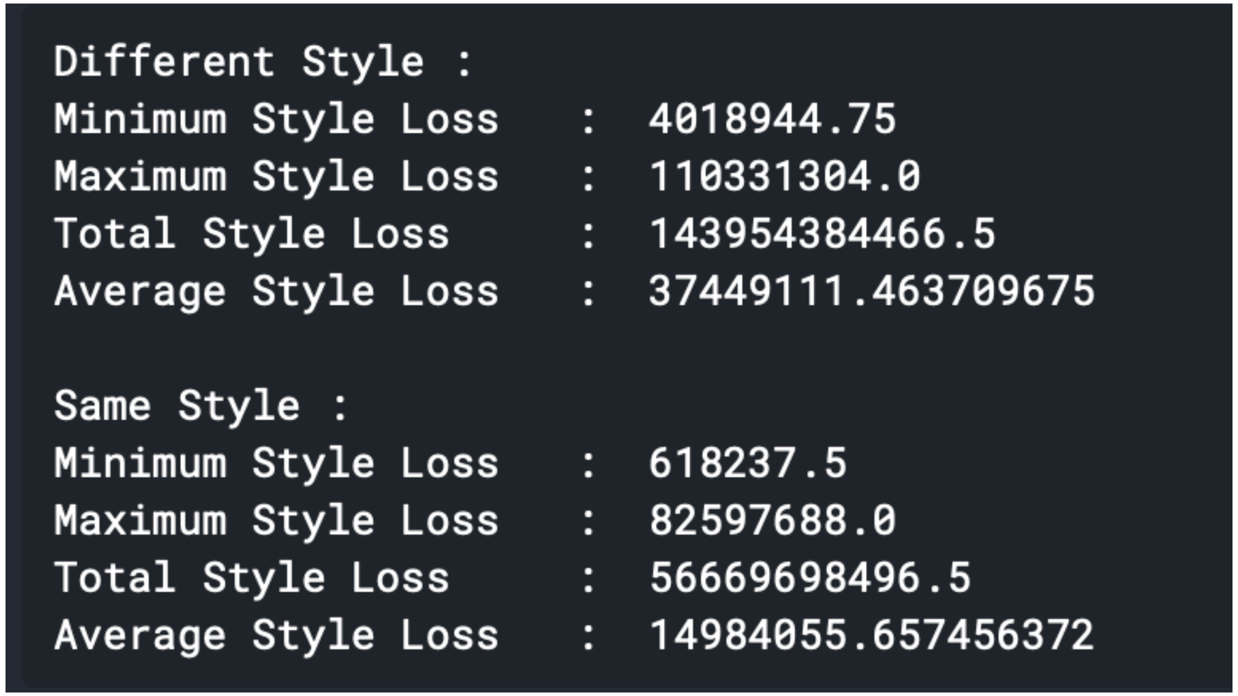


```
Different Style :
Minimum Style Loss    :  483713.25
Maximum Style Loss    :  75419184.0
Total Style Loss      :  63178097636.0
Average Style Loss    :  16435509.270551508

Same Style :
Minimum Style Loss    :  139304.84375
Maximum Style Loss    :  55072780.0
Total Style Loss      :  49482394661.59375
Average Style Loss    :  13083658.027920082
```

Ratio of average style loss = 8:10

New Style Loss Function



```
Different Style :
Minimum Style Loss    :  4018944.75
Maximum Style Loss    :  110331304.0
Total Style Loss      :  143954384466.5
Average Style Loss    :  37449111.463709675

Same Style :
Minimum Style Loss    :  618237.5
Maximum Style Loss    :  82597688.0
Total Style Loss      :  56669698496.5
Average Style Loss    :  14984055.657456372
```

Ratio of average style loss = 4:10

Test Plan and Strategy

Content loss- Difference in structure of generated image and input image

Style loss- Difference in style of the generated image and style image

Content loss

G – Generated Image
T – Target Image
S – Style Image
BS – Basic Style Image

$L_{content}(G,T) = \frac{1}{2} || G^{[l]} - T^{[l]} ||^2$

Style loss

$diff\_G = G - T$
$diff\_S = S - BS$
$GG_{ij} = \Sigma\, diff\_G_{ik}\, diff\_G_{jk}$
$SS_{ij} = \Sigma\, diff\_S_{ik}\, diff\_S_{jk}$

$L_{style}(GG,SS) = \Sigma(GG_{ij} - SS_{ij})^2$

Total loss $L_{total} = \alpha\, L_{content}(G,T) + \beta\, L_{style}(GG,SS)$

Test Plan and Strategy

Algorithm:

```
style_loss = original_loss = 0
for gen_feature, orig_feature in zip( generated_features,
original_img_features):
        original_loss += torch.mean((gen_feature - orig_feature) ** 2)

for style_features in style_features_all:
        for gen_feature, style_feature in zip(batch_size, channel, height,
        width = gen_feature.shape
                generated_diff = gen_feature - orig_feature
                style_diff = style_feature - base_feature
                GG = generated_diff.view(channel, height * width).mm(
                        generated_diff.view(channel, height * width).t())
                SS = style_diff.view(channel, height * width).mm(
                        style_diff.view(channel, height * width).t())
                style_loss += torch.mean((GG - SS) ** 2)
```

Results and Discussion

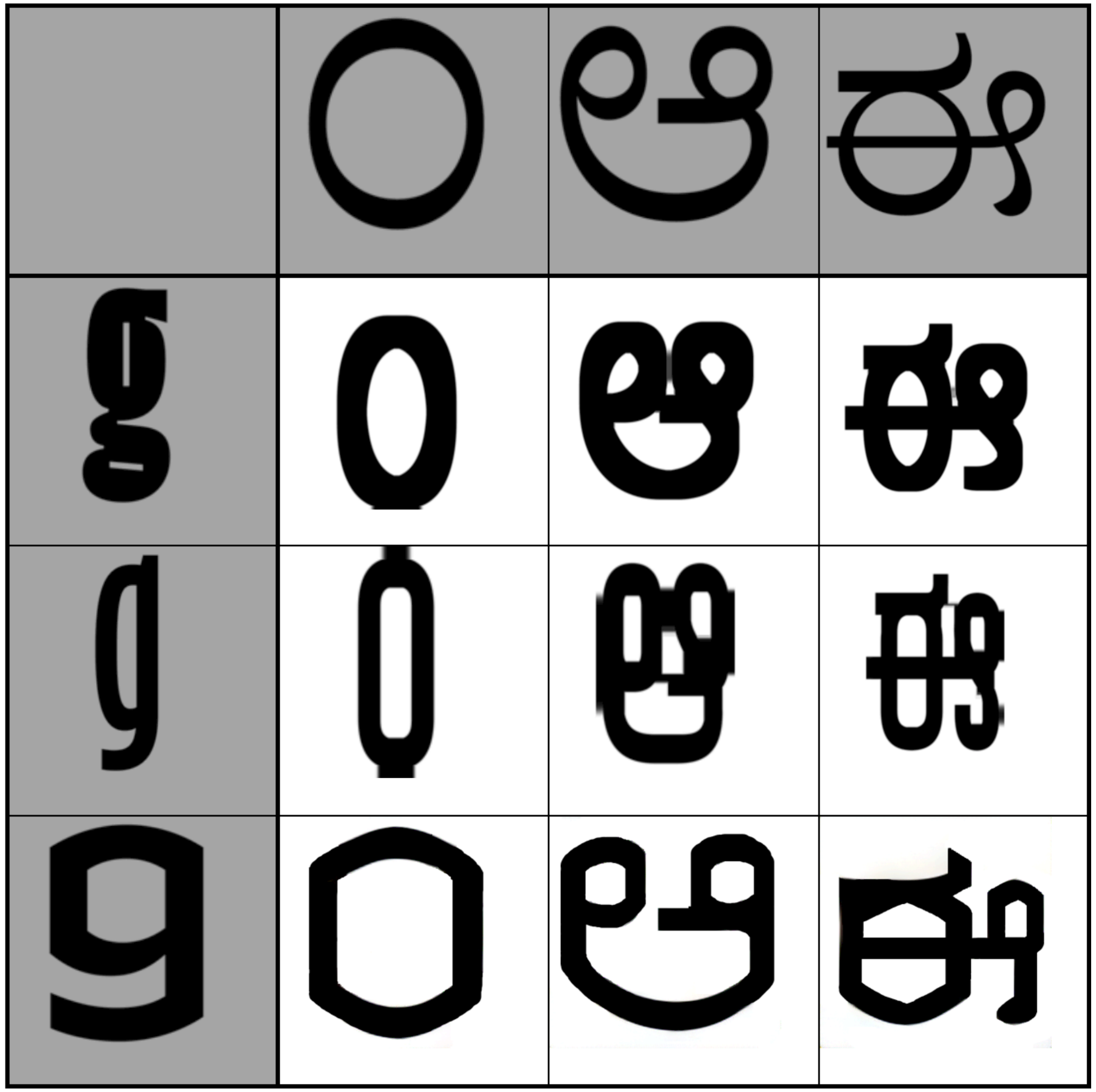
Results

Top row: Input letters

First column: Style input

1. Schmalfette
2. Picket
3. Locomo

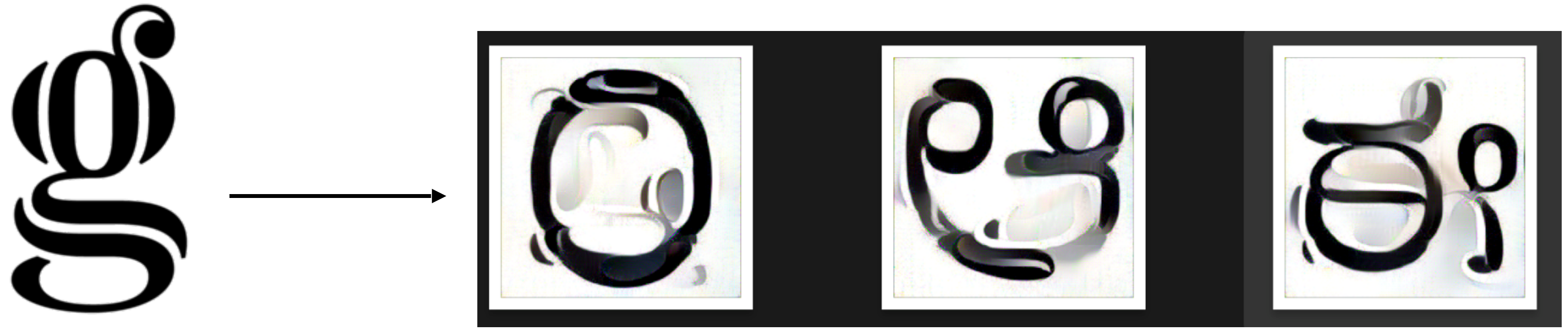
Other boxes: Generated output



Results and Discussion

Discussion

- We have **obtained results as per initial estimated plan** and hence there is **no deviation** from what we expected.
- We obtained **good accuracy** for **certain styles** as seen in the previous slide-
  - **Schmalfette**
  - **Picket**
  - **Locomo**
- **However**, for some **fonts** such as **Zina**, we could **not meet** the **required** set of **expectations**.

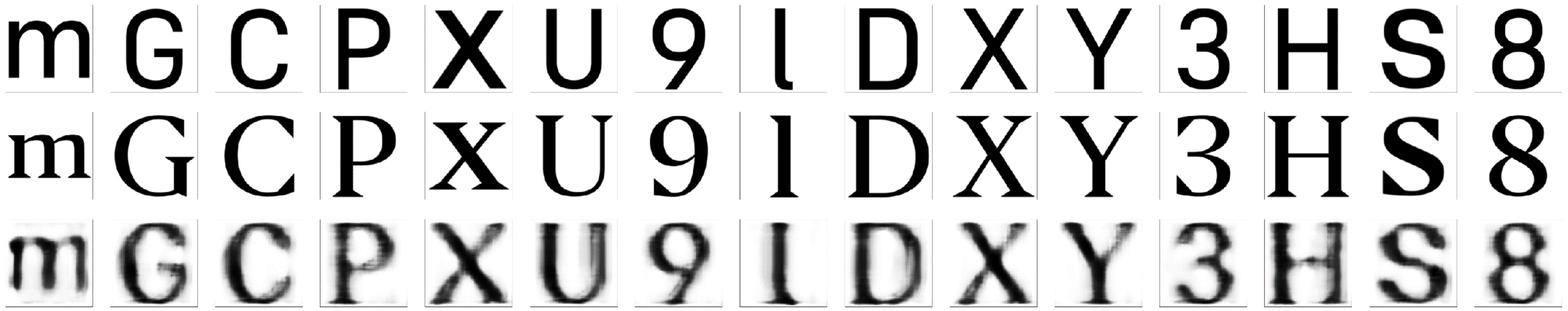


Results and Discussion

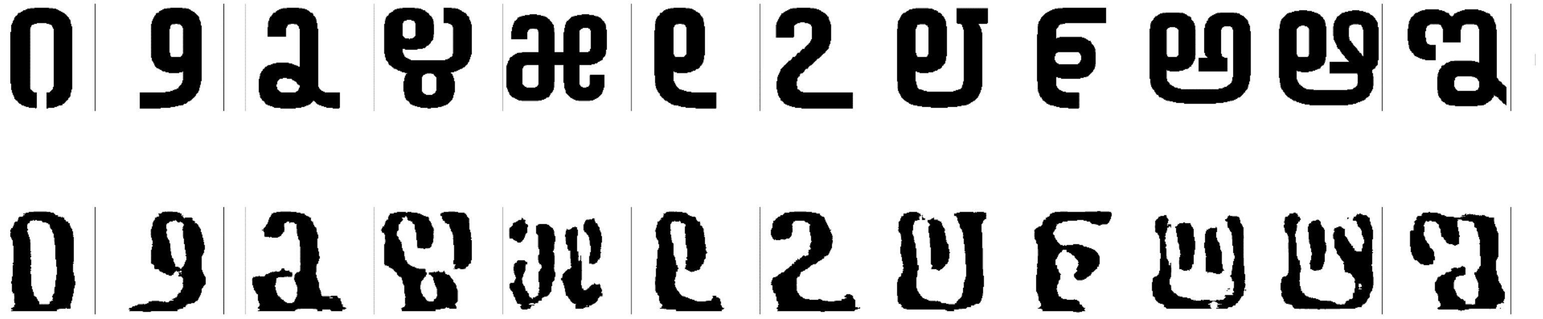
Comparison of results with Autoencoders

- We did **obtain some results** with **Autoencoders**. However, the **performance plateaued beyond** a **certain point** and hence we could not achieve the results which we were anticipating.

Training data


m G C P X U 9 l D X Y 3 H S 8
m G C P X U 9 l D X Y 3 H S 8
m G C P X U 9 I D X Y 3 H S 8

Sharpened results



Schedule

Week numbers	Planned work	Actual work
Week 1	Literature survey and definite problem formulation.	As per plan.
Week 2	Literature survey on proposed architecture.	Did literature survey + revisited previous semester’s work to understand how to integrate.
Week 3	Obtaining and cleaning dataset.	As per plan.
Week 4	Build basic model for Style Transfer.	Tried various approaches to build the basic model.
Week 5	Train basic model for Style Transfer.	Trained the various models for style transfer.
Week 6	Perform style Transfer for each alphabet of Kannada Script.	As per plan.

Schedule

Week numbers	Planned work	Actual work
Week 7	Incorporating maatras for every letter.	As per plan.
Week 8	Testing	As per plan.
Week 9	Enhanced model creation after testing	Stuck to the Neural style transfer method and enhanced the results for it.
Week 10	Fine tuning and documentation.	As per plan.
Week 11	Fine tuning and documentation.	As per plan.

Documentation

Status of the below documents:

- Project report finalized by Guide – **YES**
- IEEE format of paper ready for submission – **100% complete**
- Conferences we are you targeting -
    1. ICMLAS 2021 : Scopus-Indexed Springer International Conference on Machine Learning and Autonomous Systems.
    2. ICCCMLA 2021 : International Conference on Cybernetics, Cognition and Machine Learning Applications.
    3. ICIRCA : 3rd IEEE International Conference on Inventive Research in Computing Applications.
    4. ICCVBIC 2021 : 5th International Conference on Computational Vision and Bio Inspired Computing.

Documentation

- Video of your project – **100% complete**
- Add the Github repository link- https://github.com/saahil-jain/Font_Style_Transfer.git
- A3 size Poster of your project to be shown- **100% complete**
- All artifacts of your project uploaded in the CSE Project repository- **NO** (link was just provided)

Lessons Learnt

- Choice of loss function is extremely important and can vary based on data and not just tasks.
- On a broader aspect for a research project, background work matters a lot to define the scope of our project.
- Consider different approaches not just theoretically but also practically before opting for one.

Issues we overcame:

- In Neural style transfer, the default style loss function is used for Art images. So, it focuses on colours and shapes. But in our dataset, colours aren't considered. Hence, we developed our own loss function.
- The generated images had some noise, so we further did image processing on them to improve the quality and meet our expectations.

Conclusion and Future work

Conclusion

- The required **datasets** for both **Kannada and English** language were **obtained**.
- The **datasets** were then **cleaned** and **labelled** accordingly.
- A **basic model** was **built** and **trained** using two approaches – **Neural Style Transfer** and **Autoencoders**.
- **Results** were obatained **for both models** and **comparison** was made.
- **Neural Style Transfer provided** us with **better results** and **hence** that approach was **carried forward**.
- **Style transfer** has been **achieved** with the approach.
- **Enhancement** has been **done** using **image processing** for **intricate details** of the **Kannada language** and results were obtained.
- **Fine tuning** and **documentation** has been **completed**.

Conclusion and Future work

Future work-

Our future plans for the project is to convert these generated glyphs as fonts which can be integrated to any sort of wording software tool.

This can also work as a Proof of Concept (POC) for other Indian languages which have the same limitations in font as Kannada.

Next Steps-

Applying for conferences and getting a paper published.

References

Research Paper
[1] Samaneh Azadi, Matthew Fisher, Vladimir Kim, Zhaowen Wang, Eli Shechtman, Trevor Darrell, “Multi-Content GAN for Few-Shot Font Style Transfer”, Dec. 01, 2017. [Online]
Available: https://arxiv.org/pdf/2006.06676.pdf [Accessed Jan. 16,2021]

Research Paper
[2] Yonggyu Park, Junhyun Lee, Yookyung Koh, Inyeop Lee, Jinhyuk Lee, Jaewoo Kang, “Typeface Completion with Generative Adversarial Networks”, Dec. 13, 2018. [Online],
Available: https://arxiv.org/pdf/1811.03762.pdf [Accessed Jan. 05, 2020]

References

Research Paper

[3] Yizhi Wang, Peking University, “Attribute2Font”, May. 16, 2020. [Online], Available: https://arxiv.org/pdf/2005.07865v1.pdf [Accessed Jan. 22, 2021]

Research Paper

[4] Leon A. Gatys, Alexander S. Ecker, Matthias Bathge, “A Neural Algorithm of Artistic Style”, Sep. 02, 2015. [Online], Available: https://arxiv.org/pdf/1508.06576.pdf [Accessed Feb. 01, 2021]

Thank You