MASTER IN COMPUTER SCIENCE

Contrastive Learning for Character Detection in Ancient Greek Papyri

Advisors: Prof. Rolf Ingold,
Prof. Andreas Fischer
Supervisor: Lars Vögtlin

Master Thesis
September 6th, 2024

Vedasri Nakka
Vedasri.nakka@unine.ch

Agenda

- Introduction
- Methods
- Results
- Discussion & limitations
- Conclusion & Future work

Research Question

How effective is SimCLR contrastive learning for Greek-letter recognition tasks compared to traditional baseline methods, and which data augmentation strategies enhance its performance?

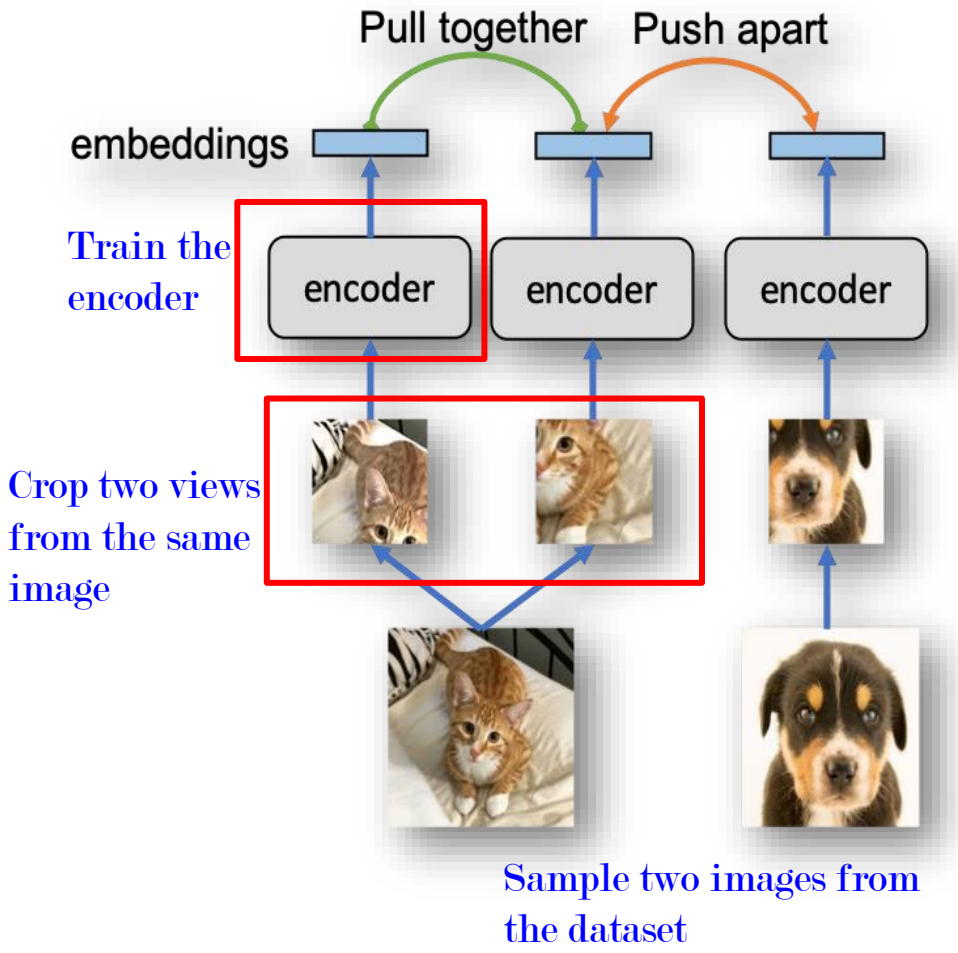
Overview of Contrastive Learning

Contrastive Learning is a **Self-Supervised Learning** technique, that teaches the model to learn embeddings of unlabeled data by

- **maximizing** the similarity between similar instances and ;
- **minimizing** the similarity between dissimilar instances.

Major Benefit:

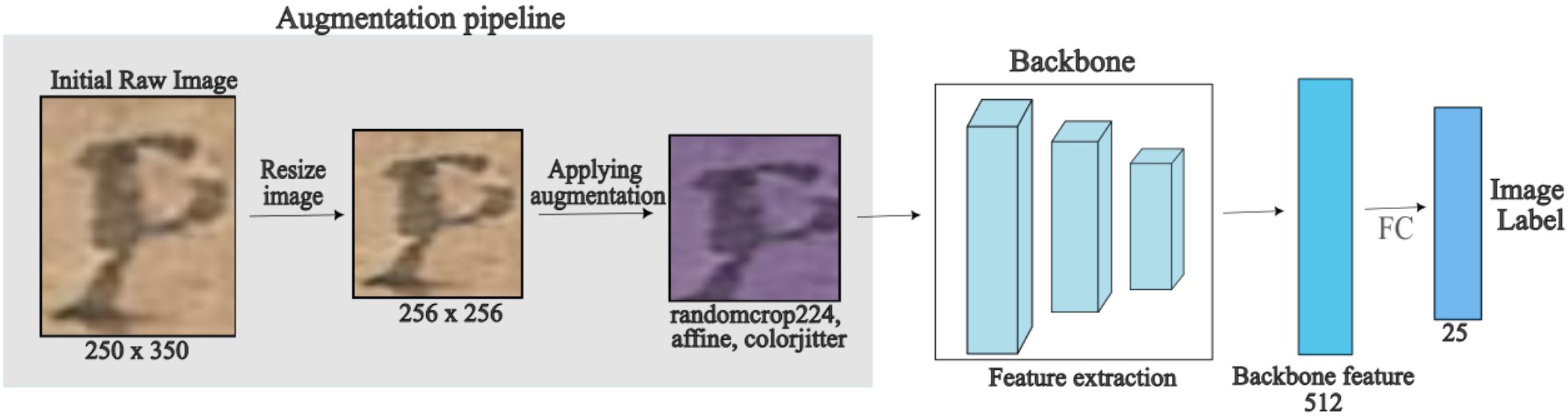
- CL do not require labels to conduct the training and thus can be pretrained on large unlabeled corpus to obtain a powerful feature encoder.



Training Strategies

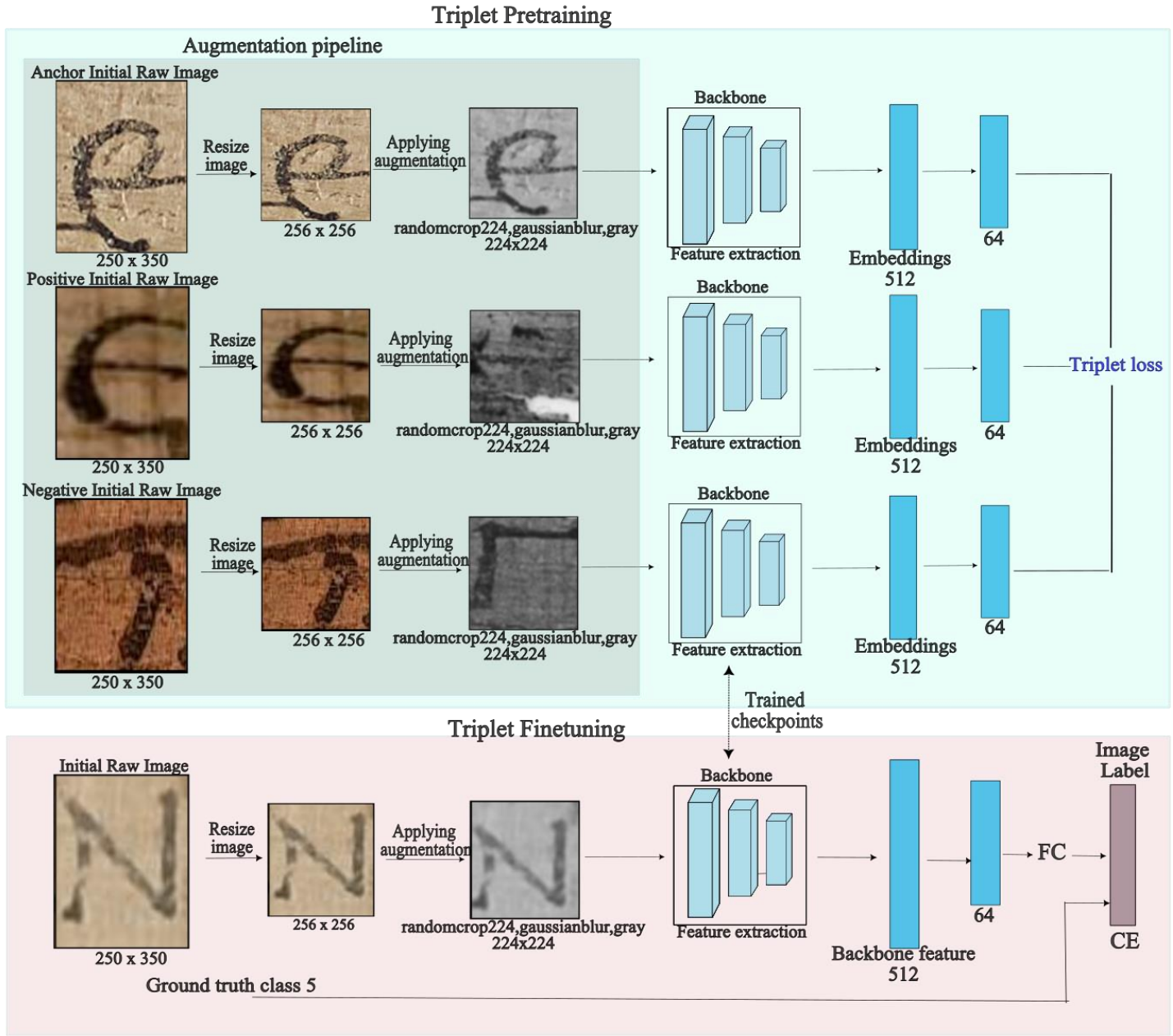
Baseline Model	Supervised learning model used for classification tasks utilizing cross-Entropy Loss.
Triplet Embedding Model	Minimize the distance between the anchor and positive image embeddings while maximizing the distance between anchor with the negative image embeddings
SimCLR Model	A self-supervised learning method using positive pairs (augmented views of the same image) and contrasts them with negative pairs (different images) to learn feature representations using Contrastive Loss.

Baseline model pipeline



- We first perform the spatial and pixel-level augmentation to obtain the final pre-processed image. We send the augmented image to the model which extracts a feature vector at the end of the backbone. The final feature vector is sent to the classification layer to get output probability.

Triplet model pipeline



Alpub dataset

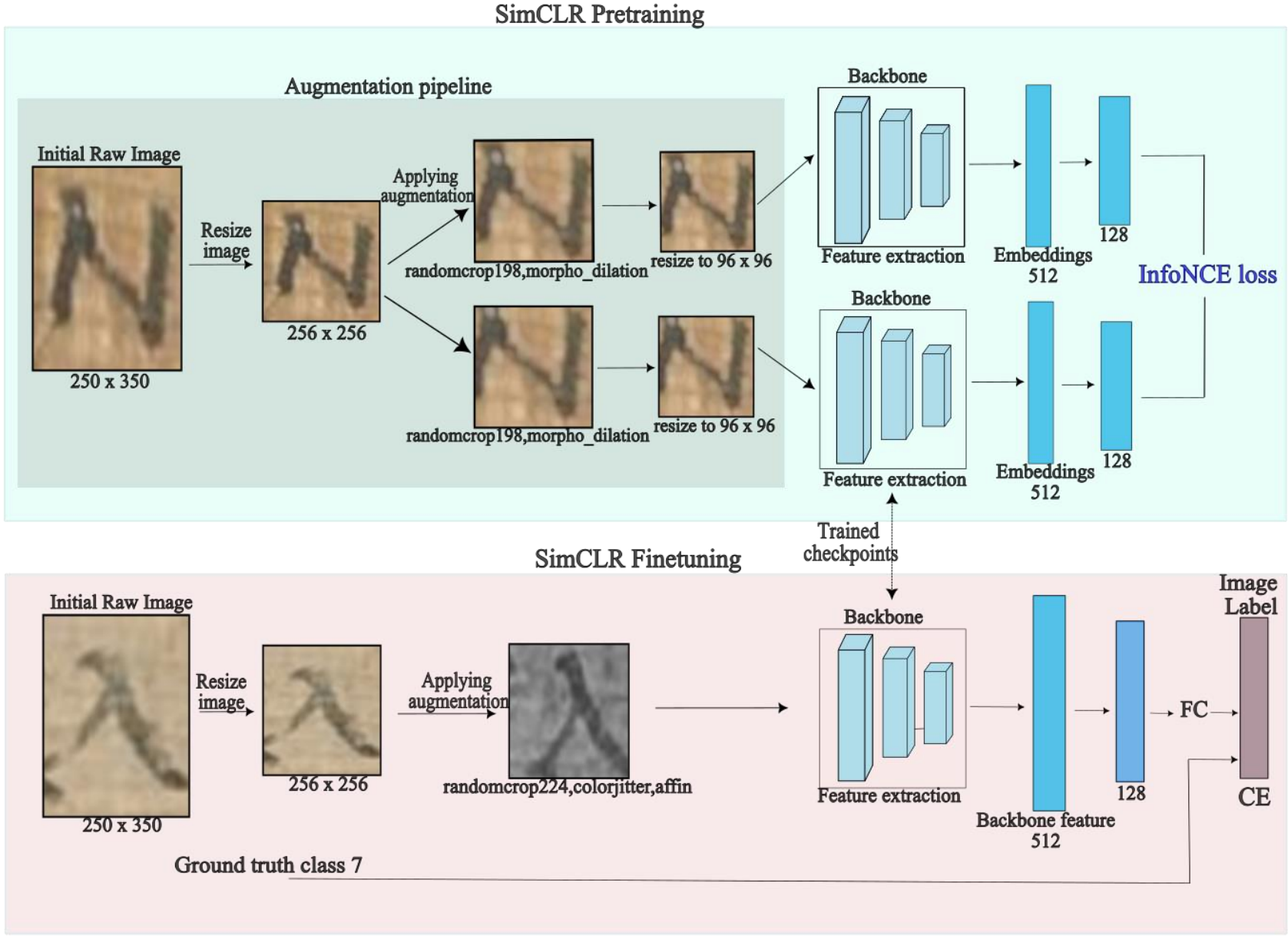
ICDAR dataset

- The top block, highlighted with a green background, represents the pretraining stage, where the model is trained on the pretraining dataset using triplet loss to learn embeddings from triplet pairs.
- In the next stage, a classification layer is added on top of the embedding layer, and the model is trained end-to-end with cross-entropy (CE) loss.

SimCLR model pipeline

Alpub dataset

ICDAR dataset



- The top block, highlighted with a green background, represents the pretraining stage, where the model is trained on the pretraining dataset using InfoNCE loss to learn embeddings from SimCLR augmented view of same image.
- In the next stage, a classification layer is added on top of the embedding layer, and the model is trained end-to-end with cross-entropy (CE) loss.

Comparison of batch data for 3 Models

Baseline model

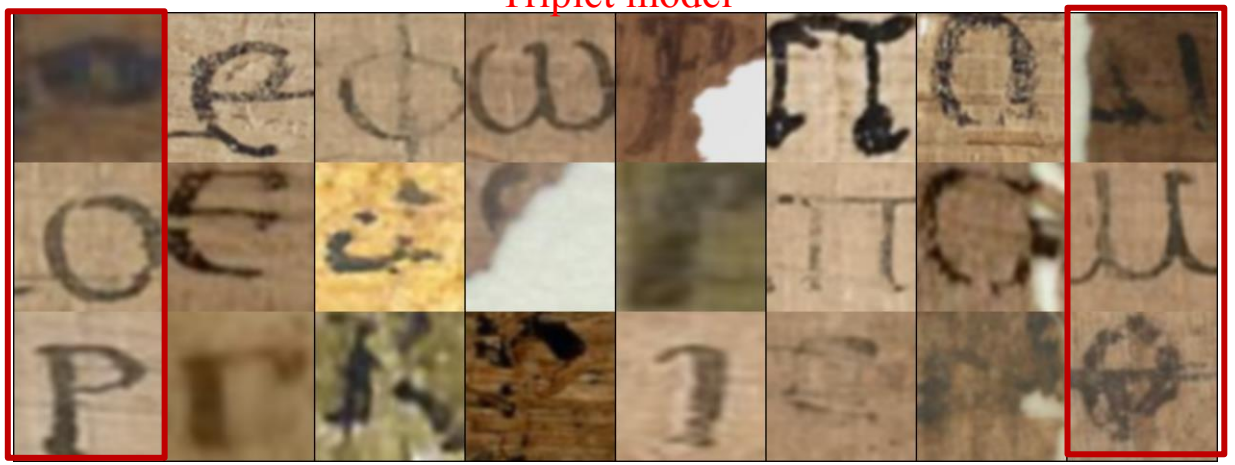
Batch of data from 16 datapoints – 4 order

Triplet model

Anchor

Positive pair

Negative pair



Batch of data from 8 datapoints (each datapoint contributes to anchor, positive, negative pairs)

The dataloader selects 8 images from the same class as the anchor images (positives) and 8 images from different classes (negatives),

SimCLR model

Image 1

Image 2

Batch of data from 8 datapoints (each datapoint contributes to pair of images)

Pair of images (top and bottom) cropped from the same image with augmentation

Datasets

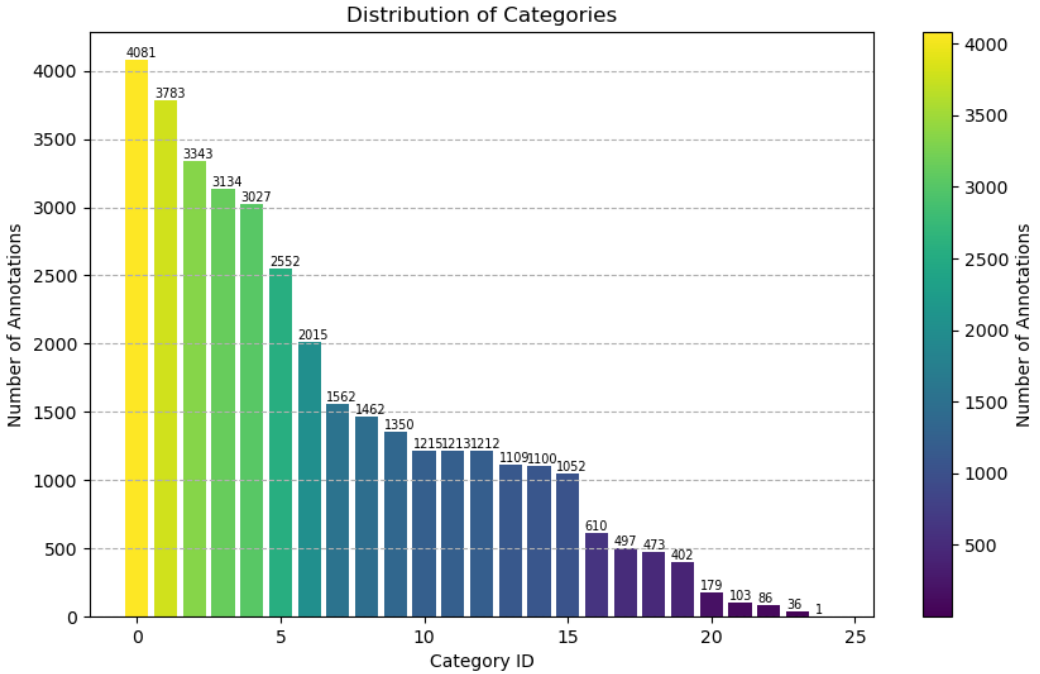
Pre-training dataset:

- AL-PUB dataset containing 205,797 cropped images
- Total of 24 Greek Letter classes

Fine-tuning dataset:

The 'ICDAR 2023 Competition on Detection and Recognition of Greek Letters on Papyri'

- **Contains** 153 training images and 34 test images of full document-level image resolution
- On cropping each Greek character from full image in dataset, we obtain around 34K cropped Greek Letter images in the training set
- Total of 25 Greek Letter classes



Cropped Greek Letters

Albumentation Library

- Albumentations is a Python library for image augmentation.
- Albumentations supports all common computer vision tasks such as classification, semantic segmentation, instance segmentation, object detection, and pose estimation.
- The library contains more than 70 different augmentations to generate new training samples from the existing data.

Types of Augmentations:

1. Pixel-level transforms: These transformations modify individual pixels in an image.

2. Spatial-level transforms: These transformations change the overall structure or geometry of the image.

List of augmentations

Pixel-level transforms

Pixel-level transforms will change just an input image and will leave any a
boxes, and keypoints unchanged. The list of pixel-level transforms:

- AdvancedBlur
- Blur
- CLAHE
- ChannelDropout
- ChannelShuffle
- ChromaticAberration
- ColorJitter
- Defocus
- Downscale
- Emboss
- Equalize
- FDA
- FancyPCA
- FromFloat
- GaussNoise
- GaussianBlur
- GlassBlur
- HistogramMatching
- HueSaturationValue
- ISONoise
- ImageCompression

Spatial-level transforms

Spatial-level transforms will simultaneously change both an input image as well
bounding boxes, and keypoints. The following table shows which additional tar

Transform	Image	Mask	BBoxes	Keypoints
Affine	✓	✓	✓	✓
BBoxSafeRandomCrop	✓	✓	✓	
CenterCrop	✓	✓	✓	✓
CoarseDropout	✓	✓		✓
Crop	✓	✓	✓	✓
CropAndPad	✓	✓	✓	✓
CropNonEmptyMaskIfExists	✓	✓	✓	✓
D4	✓	✓	✓	✓
ElasticTransform	✓	✓	✓	
Flip	✓	✓	✓	✓
GridDistortion	✓	✓	✓	
GridDropout	✓	✓		
HorizontalFlip	✓	✓	✓	✓
Lambda	✓	✓	✓	✓
LongestMaxSize	✓	✓	✓	✓
MaskDropout	✓	✓		
Morphological	✓	✓		
NoOp	✓	✓	✓	✓

Data augmentation dictionary

```python
transform_dict = {
    "default": A.Resize(height=256, width=256, p=1.0),
    "resize256": A.Resize(height=256, width=256, p=1),
    "randomcrop224": A.RandomCrop(224, 224, p=1.0),
    "randomcrop198": A.RandomCrop(198, 198, p=1.0),
    "randomcrop128": A.RandomCrop(128, 128, p=1.0),
    "randomcrop180": A.RandomCrop(180, 180, p=1.0),

    # scale hyperparam
    "morpho_dilation": A.Morphological(scale=(7, 7), operation="dilation", p=0.5),
    # scale hyperparam
    "morpho_erosion": A.Morphological(scale=(7, 7), operation="erosion", p=0.5),
    "hflip": A.HorizontalFlip(p=0.5),
    "colorjitter": A.ColorJitter(
        brightness=(0.8, 1),
        contrast=(0.8, 1),
        saturation=(0.8, 1),
        hue=(-0.5, 0.5),
        always_apply=False,
        p=0.5,
    ),
    # blur_limit
    "gaussianblur": A.GaussianBlur(
        blur_limit=(3, 7), sigma_limit=0, always_apply=False, p=0.5
    ),
    #  shift, scale, rotate
    "affine": A.ShiftScaleRotate(
        shift_limit=0.05,
        scale_limit=0.1,
        rotate_limit=30,
        border_mode=cv2.BORDER_CONSTANT,
        value=0,
        p=0.5,
    ),
    "invert": A.InvertImg(p=0.5),
    "gray": A.ToGray(p=0.5),
}
```

Composing data-augmentation given a string (Example: randomcrop224,colorjitter,invert,gray)

```python
composed_transforms.append(transform_dict["resize256"])

# add all interim transforms
if transform_types != "":
    transform_types = transform_types.split(
        ","
    )  # Split the string into a list of transform types
    for transform_type in transform_types:
        if transform_type not in transform_dict:
            raise ValueError(f"Invalid transform_type: {transform_type}")
        composed_transforms.append(transform_dict[transform_type])
else:
    assert False, "Not implemented!"

RESIZE_DIM = 96

# add all final preprocessing transforms
composed_transforms.extend(
    [
        # we resize all images to RESIZE_DIM
        A.Resize(height=RESIZE_DIM, width=RESIZE_DIM, p=1),
        A.Normalize(mean=(0.485, 0.456, 0.406), std=(0.229, 0.224, 0.225)),
        ToTensorV2(),
    ]
)

transform_type = A.Compose(composed_transforms)
test_or_val_transforms =  A.Compose(
        [
            A.Resize(height=256, width=256, p=1),
            A.CenterCrop(height=224, width=224),

            A.Resize(height=RESIZE_DIM, width=RESIZE_DIM, p=1),
            A.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225]),
            ToTensorV2(),
        ]
)
```

Resizing all datapoints to 256

Performing data augmentations pipeline.(randomcrop224,colorjitter,invert,gray)

Standard resize to 96 to preserve memory

Primary 10 augmentations

Index	Augmentation	Type	Hyperparameters
1	invert	Pixel-Level	-
2	gray	Pixel-Level	-
3	gaussianblur	Pixel-Level	blur_limit = (3, 7), sigma_limit = 0
4	colorjitter	Pixel-Level	brightness = (0.8, 1), saturation = (0.8, 1), Contrast = (0.8, 1), hue = (-0.5, 0.5)
5	resize256	Spatial-Level	-
6	randomcrop256	Spatial-Level	-
7	hflip	Spatial-Level	-
8	morpho_erosion	Spatial-Level	kernel (w, h) = (7, 7)
9	morpho_dilation	Spatial-Level	kernel (w, h) = (7, 7)
10	affine	Spatial-Level	shift_limit = 0.05, scale_limit = 0.1, rotate_limit = 30

Ray Tune Hyperparameter Optimization Library

Ray tune hyper parameter optimization library

Provide list of hyper parameter

```
# ]

choices: [
  "resize256", "resize256,morpho_erosion", "resize256,morpho_dilation", "resize256,affine",
  "resize256,colorjitter", "resize256,hflip", "resize256,invert", "resize256,gaussianblur",
  "resize256,morpho_erosion,morpho_dilation", "resize256,morpho_erosion,affine",
  "resize256,morpho_erosion,colorjitter", "resize256,morpho_erosion,hflip", "resize256,morpho_erosion,invert",
  "resize256,morpho_erosion,gaussianblur", "resize256,morpho_dilation,affine", "resize256,morpho_dilation,colorjitter",
  "resize256,morpho_dilation,hflip", "resize256,morpho_dilation,invert", "resize256,morpho_dilation,gaussianblur",
  "resize256,affine,colorjitter", "resize256,affine,hflip", "resize256,affine,invert", "resize256,affine,gaussianblur",
  "resize256,colorjitter,hflip", "resize256,colorjitter,invert", "resize256,colorjitter,gaussianblur",
  "resize256,hflip,invert", "resize256,hflip,gaussianblur", "resize256,invert,gaussianblur",
  "resize256,morpho_erosion,morpho_dilation,affine", "resize256,morpho_erosion,morpho_dilation,colorjitter",
  "resize256,morpho_erosion,morpho_dilation,hflip", "resize256,morpho_erosion,morpho_dilation,invert",
  "resize256,morpho_erosion,morpho_dilation,gaussianblur", "resize256,morpho_erosion,affine,colorjitter",
  "resize256,morpho_erosion,affine,hflip", "resize256,morpho_erosion,affine,invert",
  "resize256,morpho_erosion,affine,gaussianblur", "resize256,morpho_erosion,colorjitter,hflip",
  "resize256,morpho_erosion,colorjitter,invert", "resize256,morpho_erosion,colorjitter,gaussianblur",
  "resize256,morpho_erosion,hflip,invert", "resize256,morpho_erosion,hflip,gaussianblur",
  "resize256,morpho_erosion,invert,gaussianblur", "resize256,morpho_dilation,affine,colorjitter",
  "resize256,morpho_dilation,affine,hflip", "resize256,morpho_dilation,affine,invert",
  "resize256,morpho_dilation,affine,gaussianblur", "resize256,morpho_dilation,colorjitter,hflip",
  "resize256,morpho_dilation,colorjitter,invert", "resize256,morpho_dilation,colorjitter,gaussianblur",
  "resize256,morpho_dilation,hflip,invert", "resize256,morpho_dilation,hflip,gaussianblur",
  "resize256,morpho_dilation,invert,gaussianblur", "resize256,affine,colorjitter,hflip",
  "resize256,affine,colorjitter,invert", "resize256,affine,colorjitter,gaussianblur",
  "resize256,affine,hflip,invert", "resize256,affine,hflip,gaussianblur",
  "resize256,affine,invert,gaussianblur", "resize256,colorjitter,hflip,invert",
  "resize256,colorjitter,hflip,gaussianblur", "resize256,colorjitter,invert,gaussianblur",
  "resize256,hflip,invert,gaussianblur"

]
```

We can use hyper band schedular to efficiently search for best hyper parameter configurations. Eg: we can jointly search for different augmentations, different loss performance, different learning rates

93 combinations of augmentations

```
transform_type:
  choices: [
        "randomcrop224",

        "randomcrop224,morpho_erosion", "randomcrop224,morpho_dilation", "randomcrop224,affine", "randomcrop224,colorjitter",
        "randomcrop224,hflip", "randomcrop224,invert", "randomcrop224,gaussianblur", "randomcrop224,gray",

        "randomcrop224,morpho_erosion,morpho_dilation", "randomcrop224,morpho_erosion,affine", "randomcrop224,morpho_erosion,colorjitter",
        "randomcrop224,morpho_erosion,hflip", "randomcrop224,morpho_erosion,invert", "randomcrop224,morpho_erosion,gaussianblur",
        "randomcrop224,morpho_erosion,gray", "randomcrop224,morpho_dilation,affine", "randomcrop224,morpho_dilation,colorjitter",
        "randomcrop224,morpho_dilation,hflip", "randomcrop224,morpho_dilation,invert", "randomcrop224,morpho_dilation,gaussianblur",
        "randomcrop224,morpho_dilation,gray", "randomcrop224,affine,colorjitter", "randomcrop224,affine,hflip", "randomcrop224,affine,invert",
        "randomcrop224,affine,gaussianblur", "randomcrop224,affine,gray", "randomcrop224,colorjitter,hflip", "randomcrop224,colorjitter,invert",
        "randomcrop224,colorjitter,gaussianblur", "randomcrop224,colorjitter,gray", "randomcrop224,hflip,invert", "randomcrop224,hflip,gaussianblur",
        "randomcrop224,hflip,gray", "randomcrop224,invert,gaussianblur", "randomcrop224,invert,gray", "randomcrop224,gaussianblur,gray",

        "randomcrop224,morpho_erosion,morpho_dilation,affine", "randomcrop224,morpho_erosion,morpho_dilation,colorjitter",
        "randomcrop224,morpho_erosion,morpho_dilation,hflip", "randomcrop224,morpho_erosion,morpho_dilation,invert",
        "randomcrop224,morpho_erosion,morpho_dilation,gaussianblur", "randomcrop224,morpho_erosion,morpho_dilation,gray",
        "randomcrop224,morpho_erosion,affine,colorjitter", "randomcrop224,morpho_erosion,affine,hflip", "randomcrop224,morpho_erosion,affine,invert",
        "randomcrop224,morpho_erosion,affine,gaussianblur", "randomcrop224,morpho_erosion,affine,gray", "randomcrop224,morpho_erosion,colorjitter,hflip",
        "randomcrop224,morpho_erosion,colorjitter,invert", "randomcrop224,morpho_erosion,colorjitter,gaussianblur",
        "randomcrop224,morpho_erosion,colorjitter,gray", "randomcrop224,morpho_erosion,hflip,invert", "randomcrop224,morpho_erosion,hflip,gaussianblur",
        "randomcrop224,morpho_erosion,hflip,gray", "randomcrop224,morpho_erosion,invert,gaussianblur", "randomcrop224,morpho_erosion,invert,gray",
        "randomcrop224,morpho_erosion,gaussianblur,gray", "randomcrop224,morpho_dilation,affine,colorjitter", "randomcrop224,morpho_dilation,affine,hflip",
        "randomcrop224,morpho_dilation,affine,invert", "randomcrop224,morpho_dilation,affine,gaussianblur", "randomcrop224,morpho_dilation,affine,gray",
        "randomcrop224,morpho_dilation,colorjitter,hflip", "randomcrop224,morpho_dilation,colorjitter,invert",
        "randomcrop224,morpho_dilation,colorjitter,gaussianblur", "randomcrop224,morpho_dilation,colorjitter,gray", "randomcrop224,morpho_dilation,hflip,invert",
        "randomcrop224,morpho_dilation,hflip,gaussianblur", "randomcrop224,morpho_dilation,hflip,gray", "randomcrop224,morpho_dilation,invert,gaussianblur",
        "randomcrop224,morpho_dilation,invert,gray", "randomcrop224,morpho_dilation,gaussianblur,gray", "randomcrop224,affine,colorjitter,hflip",
        "randomcrop224,affine,colorjitter,invert", "randomcrop224,affine,colorjitter,gaussianblur", "randomcrop224,affine,colorjitter,gray",
        "randomcrop224,affine,hflip,invert", "randomcrop224,affine,hflip,gaussianblur", "randomcrop224,affine,hflip,gray", "randomcrop224,affine,invert,gaussianblur",
        "randomcrop224,affine,invert,gray", "randomcrop224,affine,gaussianblur,gray", "randomcrop224,colorjitter,hflip,invert", "randomcrop224,colorjitter,hflip,gaussianblur",
        "randomcrop224,colorjitter,hflip,gray", "randomcrop224,colorjitter,invert,gaussianblur", "randomcrop224,colorjitter,invert,gray", "randomcrop224,colorjitter,gaussianblur,gray",
        "randomcrop224,hflip,invert,gaussianblur", "randomcrop224,hflip,invert,gray", "randomcrop224,hflip,gaussianblur,gray", "randomcrop224,invert,gaussianblur,gray"
        ]
```

Results: without pretraining results on Alpub [ResNet18]

Experiment Settings

- Total cropped images: 34K
- Train-Validation-Test: 70%-15%-15%
- Models: ResNet18
- Optimizer: Adam with learning rate 0.001, 0.0003(SimCLR)
- Dataset: ICDAR dataset

Experiment	Dataset	Best Augmentation	Validation Acc.	Test Acc.
Baseline model	ICDAR	randomcrop224,morpho_erosion,morpho_dilation,gaussianblur	81.19%	80.67%
Triplet model	ICDAR	randomcrop224,morpho_dilation,affine, colorjitter	80.11%	79.16%
SimCLR model	ICDAR	randomcrop224,affine, colorjitter,gray	80.33%	80.00%

Results on ResNet-18 without pretraining on the Alpub dataset. We are presenting the best augmentations and corresponding validation and test accuracies, fine-tuned on ICDAR. The baseline model outperforms the other two methods.

Results: without pretraining results on Alpub [ResNet50]

Experiment Settings

- Total cropped images: 34K
- Train-Validation-Test: 70%-15%-15%
- Models: ResNet50
- Optimizer: Adam with learning rate 0.001, 0.0003(SimCLR)
- Dataset: ICDAR dataset

Experiment	Dataset	Best Augmentation	Validation Acc.	Test Ac.
Baseline model	ICDAR	randomcrop224,morpho_erosion, gaussianblur	80.70%	80.47%
Triplet model	ICDAR	randomcrop224,morpho_erosion, morpho_dilation,gaussianblur	79.29%	78.22%
SimCLR model	ICDAR	randomcrop224,colorjitter,gaussianblur	80.05%	79.24%

Results on ResNet-50 without pretraining on the Alpub dataset

Selecting top-4 combinations

Strategy 1: T-test based selection

Augmentations	P-value
randomcrop224,morpho_dilation,hflip	0.008096
randomcrop224,hflip,gray	0.008721
randomcrop224,invert,gaussianblur,gray	0.008746
randomcrop224,colorjitter,hflip,invert	0.010912

Ran SimCLR with **3 runs** without pretraining using all 93 augmentations. Conducted a paired t-test comparing the base augmentation, RandomCrop224, against the other 92 augmentations. Below are the top-4 augmentations with their p-values. We compute the experiment on Alpub dataset using these 4 augmentations

Strategy 2: Best average validation accuracy

Augmentations	Average Test Acc
randomcrop224,morpho_erosion,morpho_dilation,affine	79.9413
randomcrop224,morpho_dilation,affine,colorjitter	79.8826
randomcrop224,morpho_erosion,affine,colorjitter	79.8434
randomcrop224,affine,colorjitter,gaussianblur	79.7521

Averaged the values of 3 runs of SimCLR without pretraining using all 93 augmentations. Then sorted the values. These are top-4 augmentations of averaged Test accuracy results.

Results: with pretraining results on Alpub [ResNet18] using Strategy 1

Experiment Settings

- Total cropped images: 205k
- Train-Validation-Test: 70%-15%-15%
- Models: ResNet18
- Optimizer: Adam with learning rate 0.001, 0.0003(SimCLR)
- Dataset: Alpub dataset

Experiment	Dataset	Best Augmentation	Validation Acc.	Test Acc.
Baseline model	Alpub + ICDAR	randomcrop224,hflip,gray	80.49%	79.94%
Triplet model	Alpub + ICDAR	randomcrop224,morpho_dilation,hflip	78.19%	77.51%
SimCLR model	Alpub + ICDAR	randomcrop224,colorjitter, hflip,invert	77.55%	76.14%

Results on ResNet-18 with pretraining on the Alpub dataset

Results: with pretraining results on Alpub [ResNet50] using Strategy 1

Experiment Settings

- Total cropped images: 205k
- Train-Validation-Test: 70%-15%-15%
- Models: ResNet50
- Optimizer: Adam with learning rate 0.001, 0.0003(SimCLR)
- Dataset: Alpub dataset

Experiment	Dataset	Best Augmentation	Validation Acc.	Test Acc.
Baseline model	Alpub + ICDAR	randomcrop224,morpho_dilation,hflip	80.21%	79.75%
Triplet model	Alpub + ICDAR	randomcrop224,invert,gaussianblur,gray	77.90%	77.03%
SimCLR model	Alpub + ICDAR	randomcrop224,invert,gaussianblur,gray	76.90%	76.59%

Results on ResNet-50 with pretraining on the Alpub dataset

Results: with pretraining results on Alpub [ResNet18] using Strategy 2

Experiment Settings

- Total cropped images: 205k
- Train-Validation-Test: 70%-15%-15%
- Models: ResNet18
- Optimizer: Adam with learning rate 0.001, 0.0003(SimCLR)
- Dataset: Alpub dataset

Experiment	Dataset	Best Augmentation	Validation Acc.	Test Ac.
Baseline model	Alpub + ICDAR	randomcrop224,morpho_erosion,affine, colorjitter	80.68%	81.14%
Triplet model	Alpub + ICDAR	randomcrop224,morpho_dilation,affine, colorjitter	79.57%	78.88%
SimCLR model	Alpub + ICDAR	randomcrop224,morpho_erosion, affine, colorjitter	79.94%	79.18%

Results on ResNet-18 with pretraining on the Alpub dataset

Results: with pretraining results on Alpub [ResNet50] using Strategy 2

Experiment Settings

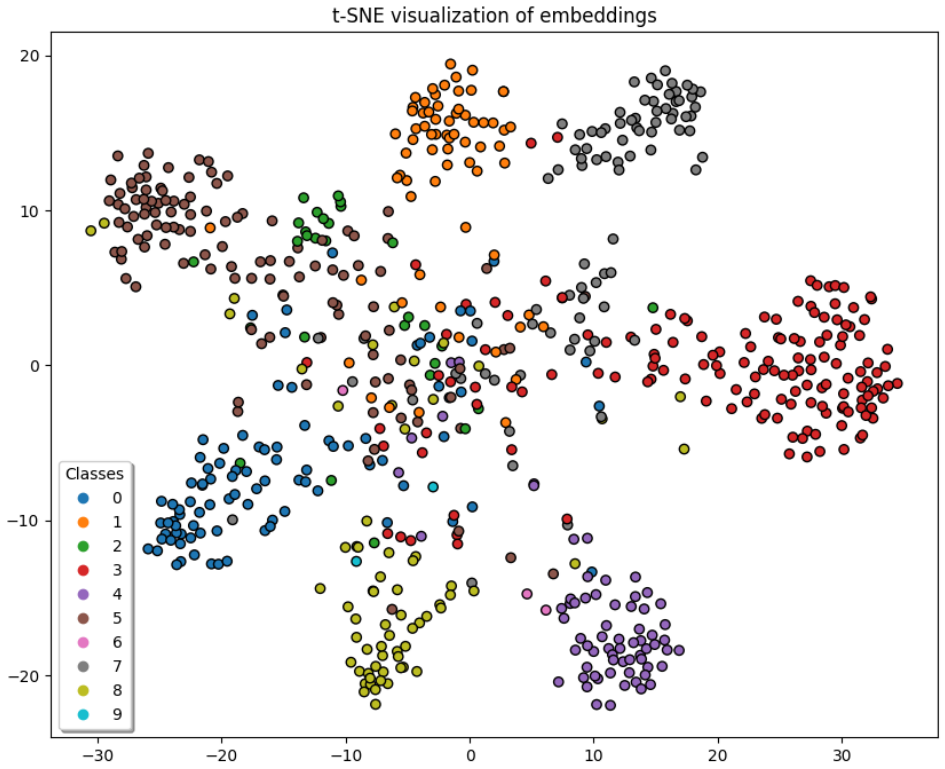
- Total cropped images: 205k
- Train-Validation-Test: 70%-15%-15%
- Models: ResNet50
- Optimizer: Adam with learning rate 0.001, 0.0003(SimCLR)
- Dataset: Alpub dataset

Experiment	Dataset	Best Augmentation	Validation Acc.	Test Ac.
Baseline model	Alpub + ICDAR	randomcrop224, affine, colorjitter, gaussianblur	81.35%	81.17%
Triplet model	Alpub + ICDAR	randomcrop224,morpho_dilation,affine, colorjitter	79.17%	78.24%
SimCLR model	Alpub + ICDAR	randomcrop224, affine, colorjitter, gaussianblur	78.68%	78.85%

Results on ResNet-50 with pretraining on the Alpub dataset

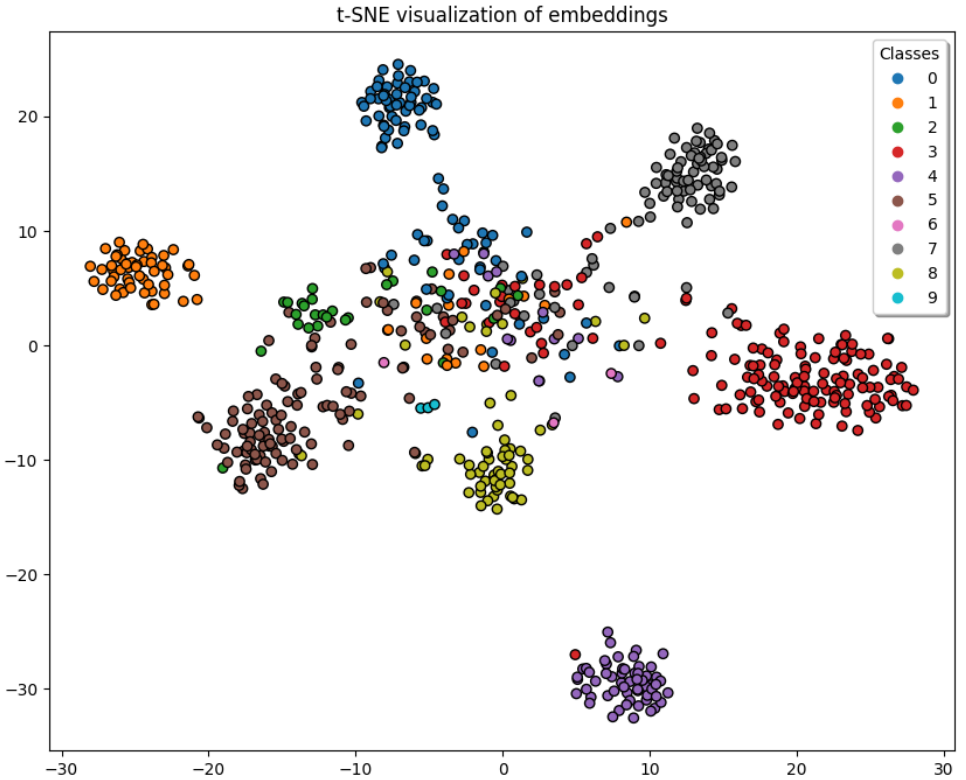
tSNE embeddings visualization of Baseline model

After pretraining embeddings



Embeddings of ICDAR dataset images after pretraining the model on Alpub dataset **for 20 epochs with CE loss** on ResNet-18

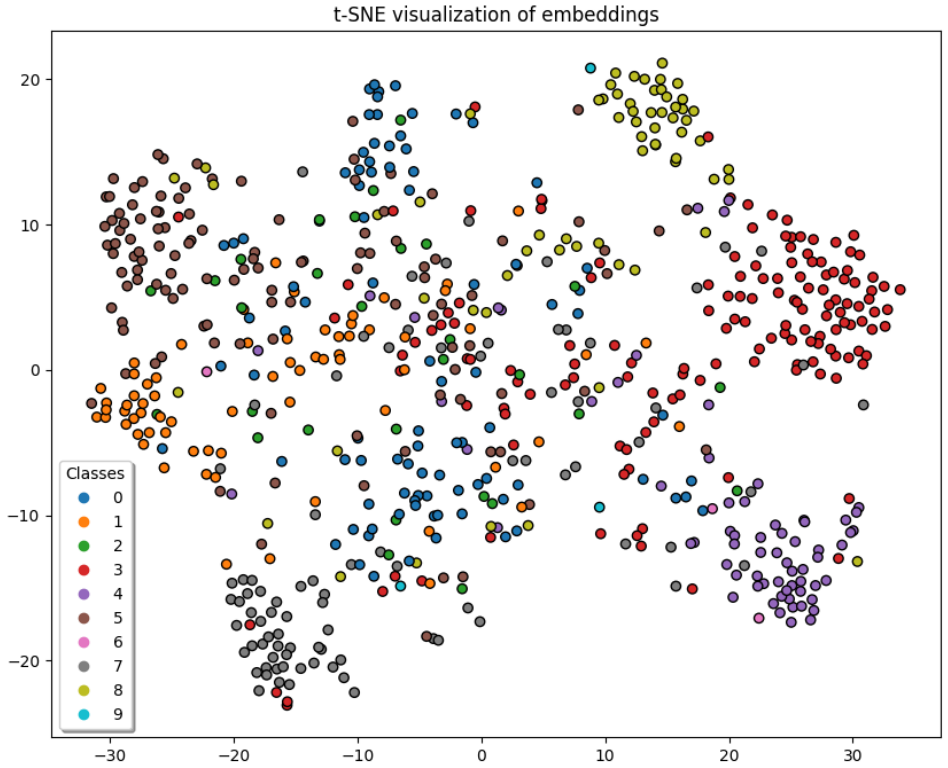
After finetuning embeddings



Embeddings of ICDAR dataset images after the model further **finetuned on all layers for 20 epochs with CE loss** on ResNet-18

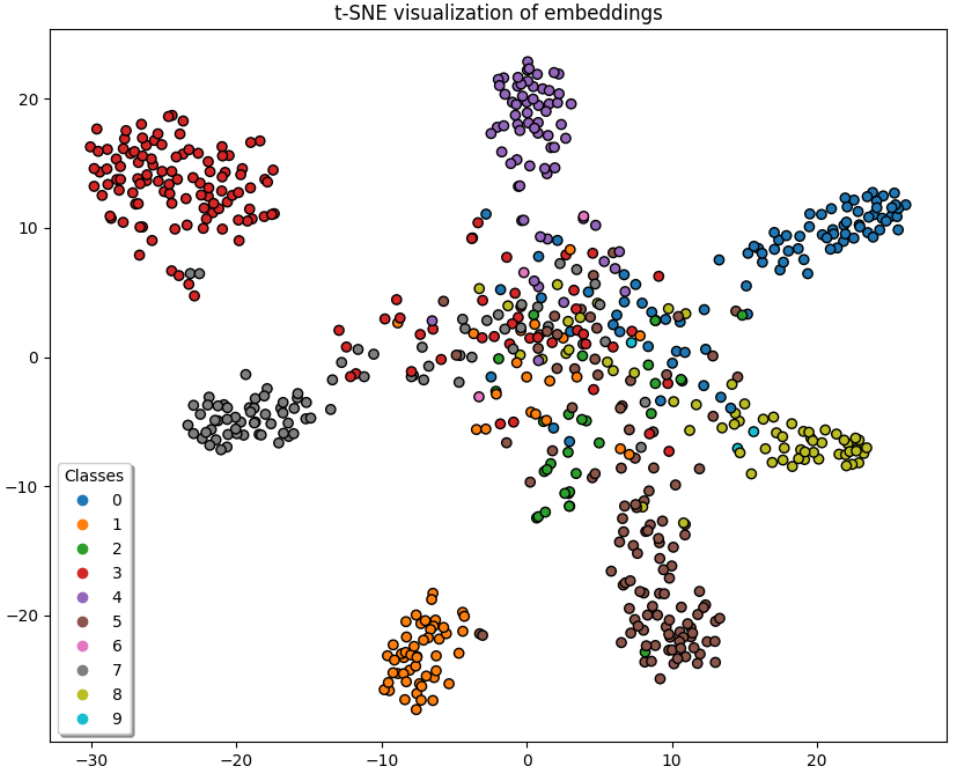
tSNE embeddings visualization of Triplet Model

After pretraining embeddings



Embeddings of ICDAR dataset images after pretraining the model on Alpub **for 20 epochs with Triplet loss** on ResNet-18

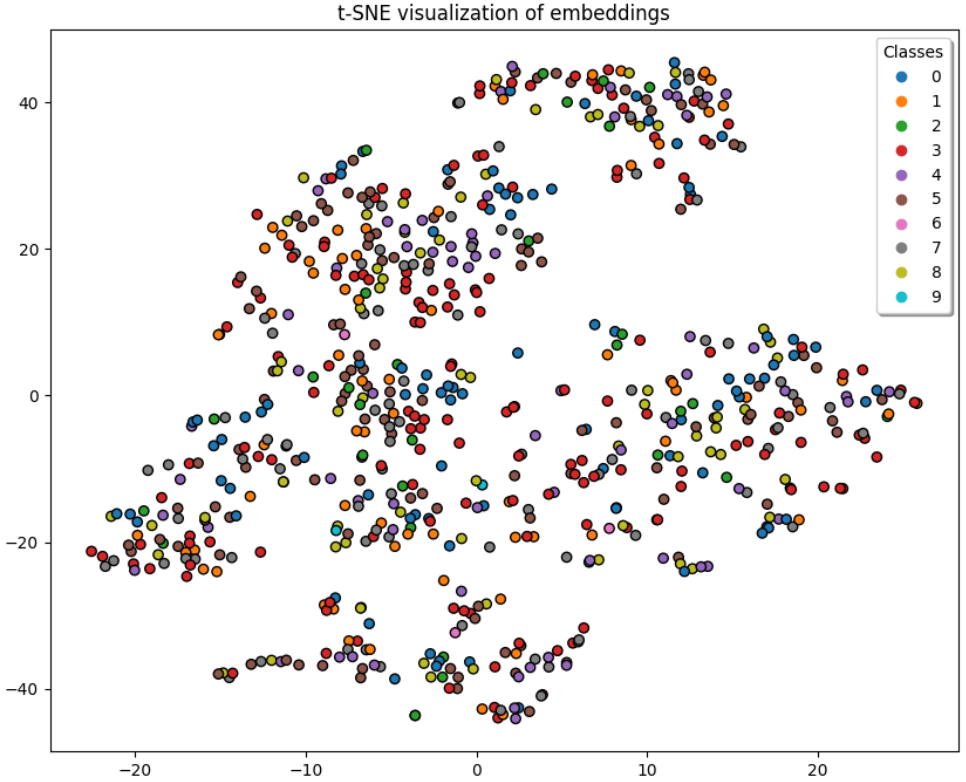
After finetuning embeddings



Embeddings of ICDAR dataset images after the model further **finetuned on all layers for 20 epochs with CE loss** on ResNet-18

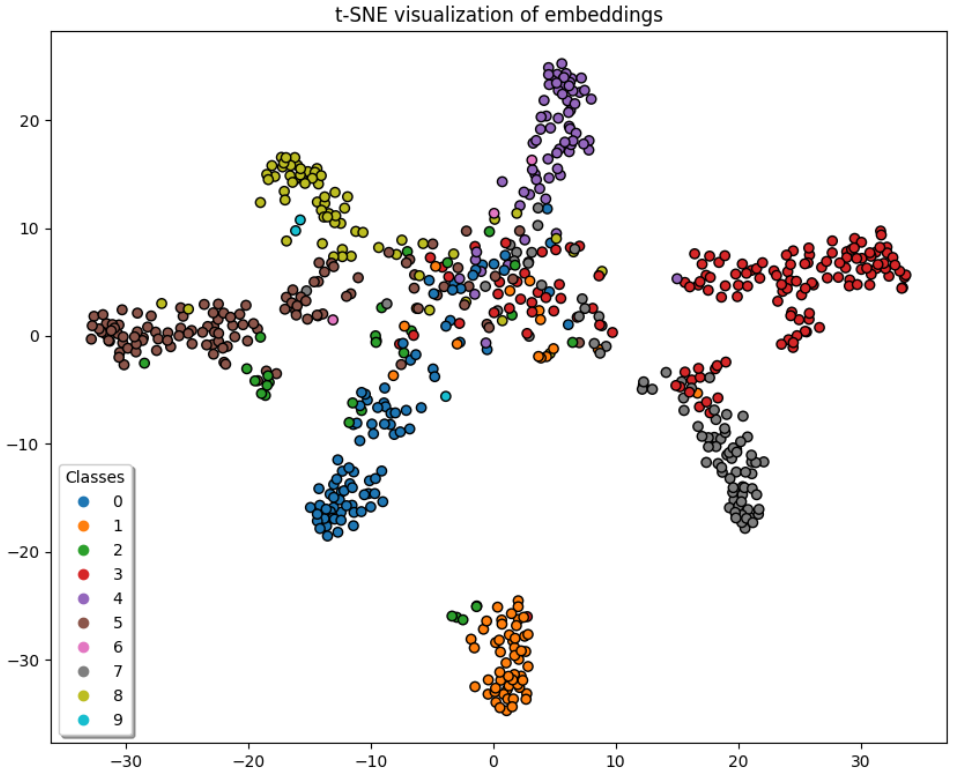
tSNE embeddings visualization of SimCLR model

After pretraining embeddings



Embeddings of ICDAR dataset images after pretraining the model on Alpub dataset **for 20 epochs with InfoNCE loss** on ResNet-18

After finetuning embeddings



Embeddings of ICDAR dataset images after the model further **finetuned on all layers for 20 epochs with CE loss** on ResNet-18

Discussion and Limitations

SimCLR Cropping Leads to Semantic Shift. We observe the two views of the image cropped from the original image with 60% area. It can be seen that this cropping scheme leads to a change in the labels.

- ❑ Hyper Parameter Tuning
- ❑ Data augmentation
- ❑ Cropping size of SimCLR
- ❑ Batch size in SimCLR
- ❑ Dataset construction

Conclusion and Future Work

- SimCLR not performing well than the traditional methods, cross-entropy and triplet loss, for letter recognition.
- Using different data augmentations, especially cropping, caused SimCLR to struggle due to changes in the meaning of the images.
- The traditional cross-entropy model consistently performed better than SimCLR and triplet loss models.
- t-SNE plots show that SimCLR failed to form well-defined clusters during pretraining.

Future Work:

- Explore alternative contrastive learning strategies.
- Tune hyperparameters and refine data augmentation techniques for improved performance and other limitations

THANK YOU