Project – 01

ArcFace: Additive Angular Margin Loss for Deep Face Recognition

[Submitted by: Group 6 – Keerthana Goka, Kunal Malhan, Sion Sharma]

Description of code

ArcFace loss function is defined as:

$$L = -\frac{1}{N}\sum_{i=1}^{N}\frac{e^{s(\cos(\theta_{y_i}+m))}}{e^{s(\cos(\theta_{y_i}+m))}+\sum_{j=1,j!=y_i}^{n} e^{s(\cos\theta_{y_i})}}$$

Which is similar to SoftMax function L_s given below:

$$L_s = -\frac{1}{N}\sum_{i=1}^{N}\frac{e^{z_i}}{\sum_{j=1}^{n} e^{z_i}}$$

ArcFace differ from SoftMax in a way that instead to $z_i = W_{y_i}^T x_i + b_{y_i}$, angular geodesic distance defined as $z_i = s(\cos(\theta_{y_i} + m))$ is used for loss computation. Below is the step-by-step explanation for the code through snippet:

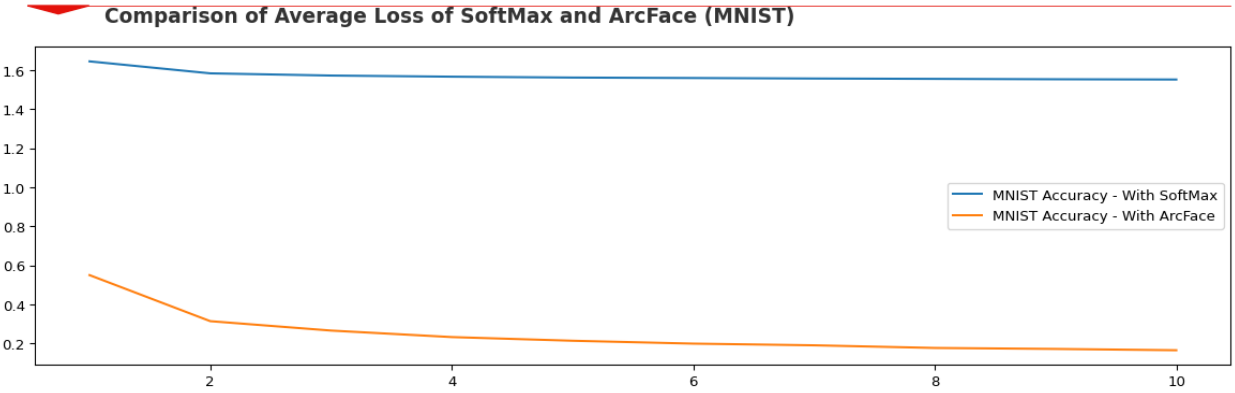
```python
class ArcFaceLoss(torch.nn.Module):
    def __init__(self, s=64.0, margin=0.25):
        super(ArcFaceLoss, self).__init__()
        self.s           = s                                  # Hyperspere radius to imrove separability
        self.margin      = margin                             # Additive angular margin to increase class separation
        self.cos_m       = math.cos(margin)
        self.sin_m       = math.sin(margin)
        self.theta       = math.cos(math.pi - margin)
        self.sinmm       = math.sin(math.pi - margin) * margin
        self.easy_margin = False

    def forward(self, logits: torch.Tensor, labels: torch.Tensor):
        index           = torch.where(labels != -1)[0]          # Extracting indices of all valid labels (ignoring -1)
        target_logit    = logits[index, labels[index].view(-1)]  # Extracting the logits corrosponding to the target class for each valid samp

        with torch.no_grad():
            target_logit.arccos_()                                          # Computing angle for target logits
            logits.arccos_()                                                # Computing angle for logits
            final_target_logit                  = target_logit + self.margin  # Adding additive angular margin
            logits[index, labels[index].view(-1)] = final_target_logit      # Replacing the target logit with additive margin
            logits.cos_()                                                   # Computing cosine of logits
        logits = logits * self.s                                            # Multiply with hypersphere scale
        return logits
```

Results

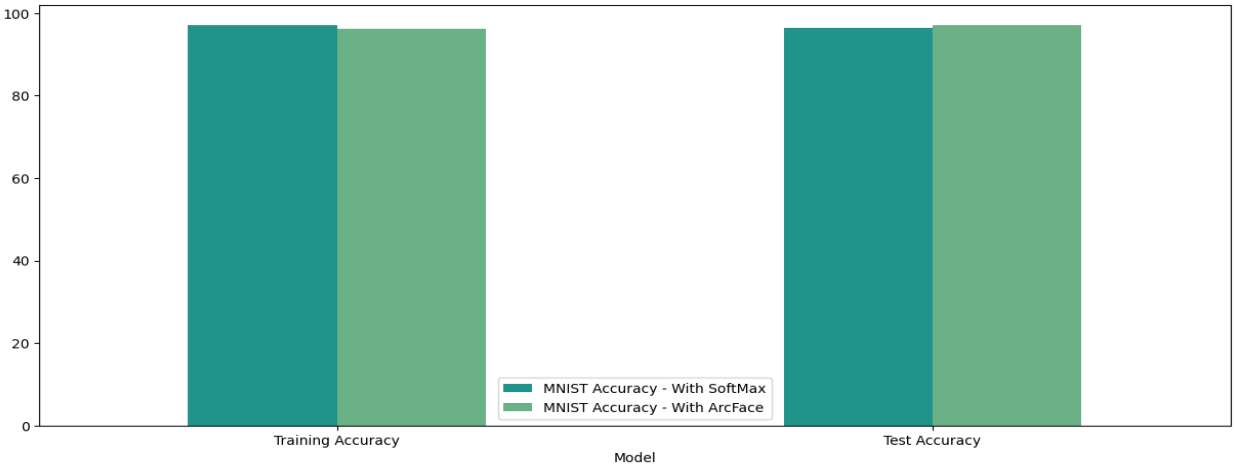
SN	Dataset	Description	Using SoftMax Loss Function	Using ArcFace Loss Function
1	MNIST Dataset	Average Loss	1.55	0.16
		Train Accuracy	97%	97%
		Test Accuracy	96%	97%
2	CIFAR10 Dataset	Average Loss	0.82	0.63
		Train Accuracy	72%	77%
		Test Accuracy	63%	80%

For MNIST Dataset:

Comparison of Average Loss of SoftMax and ArcFace (MNIST)

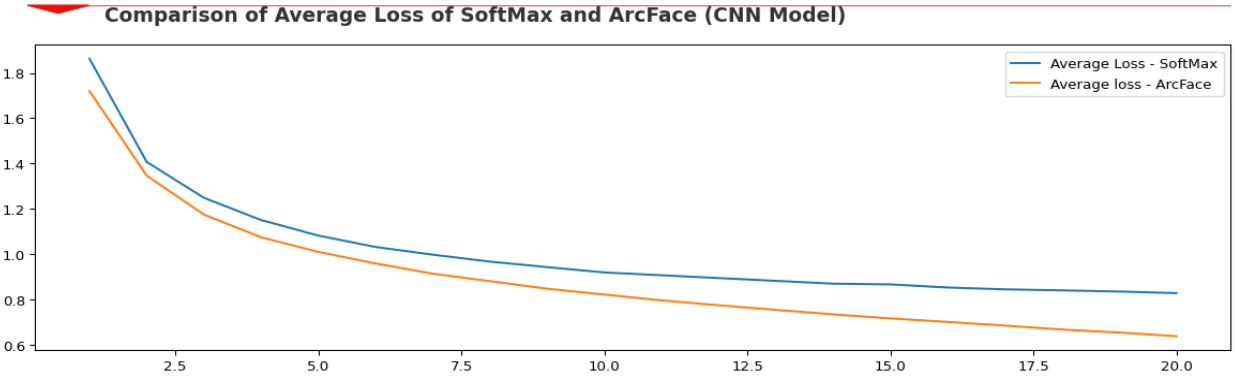


Comparison of Accuracy of SoftMax and ArcFace (MNIST)

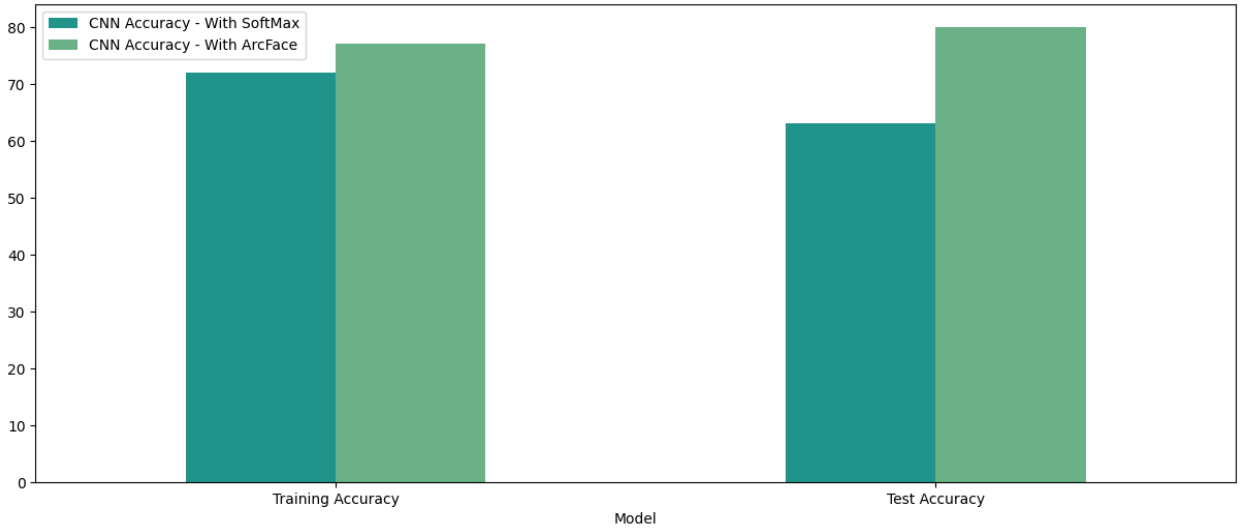


For CIFAR10 Dataset:

Comparison of Average Loss of SoftMax and ArcFace (CIFAR10)



Comparison of Accuracy of SoftMax and ArcFace (CIFAR10)



Conclusion

Although in case of MNIST dataset that is negligible increase in accuracy (both loss function shows train and test accuracy near 97%), ArcFace has shown huge improvement of both train and test accuracy in case of CIFAR10 CNN model. Training accuracy has gone up from 73% to 77% and test accuracy gone up from 63% to 80%.