7. Appendix

7.1. Derivation of the Lipschitz Constants

To ensure stability of the gradient-based method, we scale the learning rate of Γ and $\boldsymbol{A}$ by (the inverse of) approximate upper-bounds of the Lipschitz constants of the gradients, $\frac{1}{L_\Gamma}$ and $\frac{1}{L_{\boldsymbol{A}}}$, respectively. To simplify calculations, we only bound the Lipschitz constant of the gradients with respect to the matrix approximation term M, as we note this term typically dominates the Lipschitz constant of the gradient. Additionally, for notational simplicity we show the derivation for balanced classes which implies $\frac{1}{m\gamma_j} = \frac{k}{m}$.

Now, consider the function $f(\Gamma, \boldsymbol{A}) = \frac{\mu k}{2m}\sum_{j=1}^{k}\left\|\boldsymbol{Z}\mathrm{Diag}(\boldsymbol{\Pi})_j\boldsymbol{Z}^\top - \Gamma\mathrm{Diag}(\boldsymbol{A}_j)\Gamma^\top\right\|_F^2$, and note that with some simple algebra one can show the following equivalence:

$$\begin{aligned} f(\Gamma, \boldsymbol{A}) =& \frac{\mu k}{2m}\sum_{j=1}^{k}\|\boldsymbol{Z}\mathrm{Diag}(\boldsymbol{\Pi}_j)\boldsymbol{Z}^\top - \Gamma\mathrm{Diag}(\boldsymbol{A}_j)\Gamma^\top\|_F^2 \\ =& \frac{\mu k}{2m}\langle\boldsymbol{\Pi}\boldsymbol{\Pi}^\top, (\boldsymbol{Z}^\top\boldsymbol{Z})^{\odot 2}\rangle - \frac{\mu k}{m}\langle\boldsymbol{A}\boldsymbol{\Pi}^\top, (\Gamma^\top\boldsymbol{Z})^{\odot 2}\rangle + \frac{\mu k}{2m}\langle\boldsymbol{A}\boldsymbol{A}^\top, (\Gamma^\top\Gamma)^{\odot 2}\rangle \end{aligned} \tag{9}$$

where $(\cdot)^{\odot 2}$ denotes raising to the second power entry-wise.

From this, the relevant gradients are

$$\nabla_\Gamma f = -\frac{2\mu k}{m}\sum_{j=1}^{k}\left(\underbrace{\boldsymbol{Z}\mathrm{Diag}(\boldsymbol{\Pi}_j)\boldsymbol{Z}^\top\Gamma\mathrm{Diag}(\boldsymbol{A}_j)}_{:=g_j(\Gamma)} - \underbrace{\Gamma\mathrm{Diag}(\boldsymbol{A}_j)\Gamma^\top\Gamma\mathrm{Diag}(\boldsymbol{A}_j)}_{:=h_j(\Gamma)}\right) \tag{10}$$

$$\nabla_{\boldsymbol{A}} f = \frac{\mu k}{m}\left[(\Gamma^\top\Gamma)^{\odot 2}\boldsymbol{A} - (\Gamma^\top\boldsymbol{Z})^{\odot 2}\boldsymbol{\Pi}\right] \tag{11}$$

We first bound the Lipschitz constant of $\nabla_\Gamma f$. For g_j, we have

$$\frac{\|g_j(\Gamma+\Delta\Gamma) - g_j(\Gamma)\|_F}{\|\Delta\Gamma\|_F} = \frac{\|\boldsymbol{Z}\mathrm{Diag}(\boldsymbol{\Pi}_j)\boldsymbol{Z}^\top\Delta\Gamma\mathrm{Diag}(\boldsymbol{A}_j)\|_F}{\|\Delta\Gamma\|_F} \tag{12}$$

$$\leq \|\boldsymbol{Z}\mathrm{Diag}(\boldsymbol{\Pi}_j)\boldsymbol{Z}^\top\|_F\|\boldsymbol{A}_j\|_\infty, \tag{13}$$

where the inequality is due to the sub-multiplicative property of the Frobenius norm and the operator norm inequality. Now we consider h_j. Here we make the simplifying assumption that in addition to each column of Γ being a unit vector, Γ also approximately satisfies $\Gamma^\top\Gamma = \boldsymbol{I}$, which is known to be true near the globally optimal solution from the analysis of [21]. Now, we have the following approximation

$$\bar{h}_j(\Gamma) \approx \Gamma\mathrm{Diag}(\boldsymbol{A}_j)^2. \tag{14}$$

which implies,

$$\frac{\|\bar{h}_j(\Gamma+\Delta\Gamma) - \bar{h}_j(\Gamma)\|_F}{\|\Delta\Gamma\|_F} \approx \frac{\|\Delta\Gamma\mathrm{Diag}(\boldsymbol{A}_j)^2\|_F}{\|\Delta\Gamma\|_F} \tag{15}$$

$$\leq \|\boldsymbol{A}_j\|_\infty^2 \tag{16}$$

Since $f(\Gamma, \boldsymbol{A})$ has a summation over class j, we get the following approximate upper bound for the Lipschitz constant

$$L_\Gamma = \frac{2\mu k}{m}\sum_{j=1}^{k}\|\boldsymbol{Z}\mathrm{Diag}(\boldsymbol{\Pi}_j)\boldsymbol{Z}^\top\|_F\|\boldsymbol{A}_j\|_\infty + \|\boldsymbol{A}_j\|_\infty^2 \tag{17}$$

For the upper bound of the Lipschitz constant of $\nabla_{\boldsymbol{A}} f$ we simply have

$$\frac{\left\|\nabla_{\boldsymbol{A}} f\big|_{\boldsymbol{A}+\Delta\boldsymbol{A}} - \nabla_{\boldsymbol{A}} f\big|_{\boldsymbol{A}}\right\|_F}{\|\Delta\boldsymbol{A}\|_F} = \frac{\|\frac{\mu k}{m}(\Gamma^\top\Gamma)^{\odot 2}\Delta\boldsymbol{A}\|_F}{\|\Delta\boldsymbol{A}\|_F} \tag{18}$$

$$\leq \frac{\mu k}{m}\|(\Gamma^\top\Gamma)^{\odot 2}\|_F = L_A \tag{19}$$

7.2. Architecture

We utilize the following architectures for the experiments in Section 5. We use a fairly simple architecture for MNIST, and for the other datasets, we use a slightly modified version of ResNet18. Note `d` is the feature dimension.

Architecture 1 Neural Network Architecture for MNIST

```
Conv2d(in_channel=1, out_channel=32, kernel=3, stride=1)
ReLU()
Conv2d(in_channel=32, out_channel=64, kernel=3, stride=1)
ReLU()
MaxPool2d(kernel=2, stride=None)
Dropout(p=0.25)
Flatten()
Linear(12544, d)
ReLU()
Dropout(p=0.5)
Linear(d, d)
Normalize()
```

For CIFAR-10/100 and Tiny Imagenet, we use the Torchvision ResNet18 model as the featurizer, but we remove the final layer of the ResNet18 and replace it with the following to reshape the output into the desired feature dimension d. We also normalize the output at the end to fulfill the constraint that the features lie on the unit sphere in the MCR^2 objective.

Architecture 2 Reshaping Layers for ResNet18

```
Linear(512, 512, bias=False)
BatchNorm1d()
ReLU()
Linear(512, d, bias=True)
Normalize()
```

For cross-entropy experiments, we add another linear layer on top to map the output of the featurizer to logits.

7.3. Data Augmentation

We utilize the following data augmentations for the experiments in Section 5.

Transformations 1 Transformations for MNIST, CIFAR-10, CIFAR-100

```
import torchvision.transforms as transforms
TRANSFORM = transforms.Compose([
        transforms.RandomCrop(32, padding=8),
        transforms.RandomHorizontalFlip(),
        transforms.ToTensor()])
```

Transformations 2 Transformations for Tiny ImageNet

```
import torchvision.transforms as transforms
TRANSFORM = transforms.Compose([
        transforms.Resize(32)
        transforms.RandomHorizontalFlip()
        transforms.ToTensor()])
```

7.4. Additional Experiments

The experiments conducted and presented in the main body, Table 2, were only meant to compare the computational efficiency for different methods under the same conditions. The settings however were not chosen to optimize the classification performance since we did not conduct data augmentation or other training recipes normally adopted, e.g. see [22].

Here we report experimental results on CIFAR-100 and Tiny ImageNet by training with a similar training recipe adopted in [22]. Specifically, as in [22], we use PreAct ResNet-18 as the backbone and train with both cross-entropy loss and V-MCR^2. All networks are trained by SGD with momentum 0.9, and weight decay of 10^{-4}. Similar to [22], we set the learning rate to be 10^{-1} for the first half of training epochs, divide it by 10 for the next quarter of epochs, and finally divide it again by 10 for the remaining iterations. Additionally, we utilize the same transformations to augment the data. Note that the only difference we make between our strategy and the one in [22] is the choice of batch size and total number of training epochs (both specified in Section 4.3) to ensure fair comparison between CE and V-MCR^2. In addition to the training strategy, we set $\mu = 10^{-1}$ for V-MCR^2 as we find a smaller value for μ helps stabilize training in the early stage. We keep all other hyperparameters identical to ones specified in Section 4.3.

Dataset	**Objective**	**Training ΔR**	**Test Accuracy**
CIFAR-100	V-MCR^2	130.2177	0.6951
	CE	-	0.7146
Tiny ImageNet 200	V-MCR^2	134.9504	0.4189
	CE	-	0.4843

Table 3. **Comparison of classification performance.** We evaluate the training ΔR and test accuracy of V-MCR^2 after 2000 training epochs for CIFAR-100 and Tiny ImageNet. For CE, we evaluate the test accuracy at each epoch and report the highest test accuracy achieved across the 2000 epochs. [22] similarly achieved an accuracy 0.7529 for CIFAR-100 and 0.5647 for Tiny ImageNet for CE training. With V-MCR^2, we achieve a lower test accuracy. Note that the hyperparameters for V-MCR^2 such as latch frequency, dictionary size, and μ were not rigorously tuned. Additionally, due to the computational cost of the nearest subspace classification algorithm, we only evaluate the test performance at the 2000 epoch mark.