Bukovinian Math. Journal. 2026, **14** (1) , 54–69 Буковинський матем. журнал 2026, Т.14, №1, 54–69

DOI: https://doi.org/10.31861/bmj2026.01.06

Lukashov D.S.

HYPERSPHERICAL MIXED PROTOTYPES NETWORKS

This paper introduces hyperspherical mixed prototype networks. The key difference compared to hyperspherical prototype networks is mixing class prototypes and refined optimization objectives. This work proposes mixing data samples and the corresponding class prototypes in their respective spaces. In this case, the objective is to maximize the cosine similarity between a mixed sample and its corresponding mixed prototype. The other proposed objective is to minimize the cross-entropy between the dot product of a mixed sample with the original prototypes and the corresponding vector representing the proportion of each prototype in the mixture. The experiments show a performance improvement in the visual classification task compared to the baseline hyperspherical prototype networks for both optimization objectives. In addition, the results presented in this work outperform those given in the original hyperspherical prototype networks paper. Another key finding is that, when mixing is used, maximizing similarities as an optimization objective results in better accuracy and much better intraclass clustering of embeddings around their respective prototypes than cross-entropy.

Key words and phrases: Artificial intelligence, Neural networks, Deep learning, Computer vision.

Kharkiv National University of Radio Electronics, Khariv, Ukraine (Lukashov D.S.)
e-mail: *dmytro.lukashov@nure.ua (Lukashov D.S.)*

1 Introduction

Data augmentation is one of the most essential techniques for training computer vision models, especially classifiers. Many transformations change an image without altering its relation to the corresponding class, for example, rotation, flipping, color jittering, adding Gaussian noise, etc. However, another class of transformations appeared not so long ago and showed significant performance improvements. That class of transformations can be broadly called mixing. In general, mixing augmentations take a mini-batch of training samples and combine them in some way to produce a new mini-batch of the same size. Two key differences distinguish mixing augmentations from conventional. The first is that mixing usually produces samples that do not correspond to any single class but to a mixture of different classes in different proportions. The second key difference is that the result of mixing is stochastic in nature. The samples to mix are chosen randomly from a mini-batch. Moreover, mini-batches themselves are constructed from randomly selected samples. All this randomness allows for

УДК 004.85
2020 *Mathematics Subject Classification:* 68T07.

© Lukashov D.S., 2026

much greater exploration of input space, which, as evidence suggests [1—8], positively impacts the performance of networks.

When it comes to hyperspherical prototype networks [9] (and prototype networks in general), they can use conventional image augmentations as is because the relationship between the altered sample and its class (hence, the class prototype) is preserved. However, it is not immediately obvious how to adapt this class of networks to training on samples with mixed labels. The straightforward answer is to consider the results of the dot products of the embedding of a sample and the prototypes as logits and to apply cross-entropy, similar to [10]. The not-so-straight answer is to modify the loss function introduced in the original HPNs paper to accommodate mixed samples. It is crucial to consider both functions because one may have a better outcome than the other, and this is so far unknown because there is no direct comparison available between them.

This article addresses the problems described above. First, it introduces the idea of mixed prototypes, which allows for adapting the loss introduced in the original HPNs paper to samples with mixed class labels. This is done by linearly combing the prototypes corresponding to the mixture's classes. Second, this work runs a series of experiments to determine whether mixing augmentations affect the outcome of HPNs training. The experiments also compare the influence of the two optimization objectives on performance in cases without and with mixing augmentations.

2 Related Work

Two features define the concept of hyperspherical prototype networks: embeddings are normalized to have unit length, and the prototypes are chosen apriori [9, 10]. They are kept unchanged throughout the lifetime of a network. Under this definition, the space, to which input vectors are projected, is naturally constrained to some small subspace of $\mathbb{R}^n$. Some works use the same basic idea but different hypersrufaces, such as, for example, hyperbolic hypersurfaces [11] or various regular polytopes (d-simplex, d-cube, etc.) [12, 13]. However, the vast majority of works either do not normalize sample/prototype embeddings, do not use fixed prototypes, or relax both requirements simultaneously [14—20]. All these works share two unifying characteristics: they do not explore how incorporating mixing augmentations can impact the performance (except for [16]), and they exclusively rely on cross-entropy as the loss to minimize (except for [9]).

Mixing augmentations effectively improves the classification performance of conventional neural network-based classifiers [1—8]. So, it is natural to assume that at least some improvements must be observed when these augmentations are applied to prototype networks. As mentioned above, only [16] uses some form of mixing, which is done in the embedding space. As the results presented in that paper suggest, their approach improves the network's performance, which is promising for mixing in the context of hyperspherical prototype networks.

The reliance of all the methods exclusively on cross-entropy is fascinating but expected. This loss function is a proven technique for training classifiers, so it makes sense to rely on it unconditionally. However, there is not much information on cross-entropy behavior in the

context of prototype (especially hyperspherical prototype) networks compared to other loss functions because the only alternative is presented in [9]. As [10] shows, the cross-entropy can outperform the loss function of [9]. However, the difference is minimal (for balanced datasets). So, there remains the possibility that, without any mixing, the loss proposed in [9] leads to a better or comparable performance than that of [10]. As for the case where mixing augmentations are added, it is impossible to make any predictions since no experiments have been performed so far.

3 Mixed prototypes

Let us denote a dataset for a classification task as $\mathcal{D} = \{(\mathbf{x}_i, y_i) | i \in \{1, 2, ..., S\}\}$ where $\mathbf{x}_i \in \mathbb{R}^d$ is an input vector, $y_i \in \{1, 2, ..., N\}$ is the corresponding label, S is the number of samples, and N is the number of classes. For each class, choose a fixed point $\mathbf{p}_i \in \mathbb{R}^d$ called a prototype and call a set of prototypes $\mathcal{P}$. The prototypes are chosen so that $||\mathbf{p}_i||_2 = 1$ and are positioned relative to each other so that they are an approximate solution to the corresponding Thomson problem [21] (the distance between two prototypes i, j is defined as $r_{ij} = 1 - \frac{\mathbf{p}_i^T \mathbf{p}_j + 1}{2} = \frac{1 - \mathbf{p}_i^T \mathbf{p}_j}{2}$). d is the prototype's space dimensions. Also, denote a function h parametrized by some set of weights $\mathcal{W}$ that takes $\mathbf{x}_i$ as input and produces a vector in $\mathbb{R}^d$ as output. Finally, let us name $\frac{h(\mathbf{x}_i; \mathcal{W})}{||h(\mathbf{x}_i; \mathcal{W})||_2}$ as $\tilde{\mathbf{x}}_i$

With the above definitions, it is possible to define two loss functions to be optimized by tuning $\mathcal{W}$. The first one is defined in the original hyperspherical prototype networks paper [9]

$$\mathcal{L} = \sum_{i=1}^{S} (1 - \tilde{\mathbf{x}}_i^T \mathbf{p}_{y_i})^2. \tag{1}$$

It directly minimizes the cosine dissimilarity between the embeddings of data samples and the prototypes corresponding to the samples' labels. The alternative objective, which is presented in [10] can be expressed as

$$\mathcal{L} = -\frac{1}{S} \sum_{i=1}^{S} \log\left(\frac{e^{\tilde{\mathbf{x}}_i^T \mathbf{p}_{y_i}}}{\sum_{j=1}^{N} e^{\tilde{\mathbf{x}}_i^T \mathbf{p}_j}}\right). \tag{2}$$

This loss function tries to "pull" embeddings closer to their correct prototypes and "push" them away from the incorrect ones.

The loss functions defined above only work when the samples belong to exactly one class (i.e., they have hard labels). Therefore, it is necessary to generalize the two loss functions to allow samples to be mixtures of different classes (i.e., to have soft labels).

Equation (1) minimizes the dissimilarity between a sample and its corresponding prototype. However, if a sample is a mixture of several classes, there is no single prototype to which it makes sense to minimize the dissimilarity. Fortunately, it is possible to construct a point in the prototype space that correctly represents the mixed sample by linearly combining the constituent prototypes using the probabilities that a sample belongs to the classes. Formally, let us replace the class index y_i by a vector $\mathbf{y}_i$ in $\mathbb{R}^N$ where its j-th component corresponds to a probability of $\mathbf{x}_i$ belonging to class j (in the case of hard labels, $\mathbf{y}_i$ is the one-hot encoding

of a class). Define a matrix $\mathbf{P}$ such that the k-th column is occupied by $\mathbf{p}_k$, then $\frac{\mathbf{P}\mathbf{y}_i}{||\mathbf{P}\mathbf{y}_i||_2}$ is a point on the prototype hypersphere (called a mixed prototype) to the vicinity of which the mixed sample must be mapped. Equation (1) then becomes

$$\mathcal{L} = \sum_{i=1}^{S}(1 - \tilde{\mathbf{x}}_i^T \frac{\mathbf{P}\mathbf{y}_i}{||\mathbf{P}\mathbf{y}_i||_2})^2 \tag{3}$$

As for Equation (2), using the definitions from the previous paragraph, it is possible to bring it back to its more general form

$$\mathcal{L} = -\frac{1}{S}\sum_{i=1}^{S}\mathbf{y}_i^T \log(\frac{e^{\mathbf{P}^T\tilde{\mathbf{x}}_i}}{\sum_{j=1}^{N} e^{\tilde{\mathbf{x}}_i^T\mathbf{p}_j}}), \tag{4}$$

where log and exponent are applied component-wise to vectors.

Having established the objectives that allow training hyperspherical mixed prototype networks, it is now necessary to define how to mix samples and the corresponding class labels. A general mixing operation m can be described as a function that takes a list of different input samples $\mathbf{X}$ and combines them in some way (for example, for images, it can be CutMix, MixUp, etc.). m outputs a generated sample along with a vector representing a proportion of each class in the output (the values are greater than or equal to zero and normalized to sum to 1). This vector is calculated as a weighted sum of the input samples' class vectors weighted by the proportion of each input sample's presence in the generated one.

As a summary of everything described in this section, the complete training procedure for the hyperspherical mixed prototype approach for a single batch of training samples $\mathcal{B} \subset \mathcal{D}$ is stated as Algorithm 1. sample_loss function corresponds to Equations (3) or (4) with $S = 1$.

Algorithm 1 HMPNs' training procedure for a single training batch

Require: $\mathcal{B}$, a number k of samples to mix to produce a new sample, m, $h(\cdot;\mathcal{W})$, $\mathbf{P}$.

Ensure: $h(\cdot;\mathcal{W})$ is optimized to embed data samples close to their corresponding class prototypes.

$\mathcal{L} \leftarrow 0$

for $i \leftarrow 1; i \leq |\mathcal{B}|; i \leftarrow i+1$ **do**

$\mathcal{X} \leftarrow k$ samples from $\mathcal{B}$ chosen by the procedure specified by m

$\mathbf{X} \leftarrow (\mathbf{x}_i | (\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{X})$

$\mathbf{Y} \leftarrow \begin{bmatrix} | & | & & | \\ \mathbf{y}_1 & \mathbf{y}_2 & ... & \mathbf{y}_k \\ | & | & & | \end{bmatrix}$, where $\mathbf{y}_i \in \{\mathbf{y}_j | (\mathbf{x}_j, \mathbf{y}_j) \in \mathcal{X}\}$

$\mathbf{x}, \mathbf{y} \leftarrow m(\mathbf{X}, \mathbf{Y})$

$\tilde{\mathbf{x}} \leftarrow \frac{h(\mathbf{x};\mathcal{W})}{||h(\mathbf{x};\mathcal{W})||_2}$

$\mathcal{L} \leftarrow \mathcal{L} + \text{sample\_loss}(\tilde{\mathbf{x}}, \mathbf{y}, \mathbf{P})$

end for

if sample_loss is cross-entropy **then**

$\mathcal{L} \leftarrow \frac{\mathcal{L}}{|\mathcal{B}|}$

end if

compute the gradients of $\mathcal{L}$ with respect to $\mathcal{W}$

update $\mathcal{W}$ using the computed gradients and the chosen optimizer.

4 Experiments

The experiments are performed on CIFAR 10, CIFAR 100 [22], and Tiny ImageNet [23] datasets because they are standard benchmarks for the visual classification task. They offer progressively increasing task complexity with enough data variety (especially Tiny ImageNet) to make some claims about the results' scalability. However, they are still small enough in terms of the number of samples and image resolution to make quick training and evaluation possible with limited available hardware resources.

For each dataset, each model is initialized from the same weights that were randomly selected and trained for 200 epochs using the Adam [24] optimizer with a constant learning rate of 0.001 (other settings correspond to the current Pytorch defaults). The images are augmented with RandAugment [25] during training. This augmentation is applied for the models without and with mixed prototypes. Mixing is applied on top. For RandAugment, the number of operations is set to 3, the magnitude is set to 9, and the number of magnitude bins is set to 31. The random seeds in Python and Pytorch are fixed; the same applies to all the runs. The batch size is set to 128.

Cutmix [1] and Mixup [2] are used as the mixing strategies because they are proven to improve the accuracy of visual classifiers substantially. For each batch, a random decision is made whether to apply one or the other (with equal probabilities). All other parameters correspond to the current Pytorch defaults.

A linear layer is added to each base model to allow for negative embedding values (which would not have been possible due to the ReLU activation function).

The baseline models are those that use equations (1) and (2). Mixed prototype models are those that use equations (3) and (4). The models corresponding to equations (1) and (3) are termed dissimilarity models/objectives/losses, those that use (2) and (4) are termed MCE models/objectives/losses.

The entire code base is available on github[1].

4.1 CIFAR 10

For this dataset, the images are kept at their native resolution of 32x32, and the downscaled version of the ResNet [26] model is used (the same as the paper's [26] authors use for CIFAR 10 with $n = 9$). Embedding dimensions d are set to 64.

The comparison of training losses can be observed in Figure 1. As expected, when Cutmix and Mixup are applied, the losses become more unstable. However, this leads to consistently better validation accuracy and intraclass similarity, as shown in Figure 2. The clear pattern also emerges. There is a consistent difference in terms of intraclass similarities. And this behavior is identical without mixing and with it. Summed squared dissimilarity loss leads to better clustering around the prototypes, implying a higher network robustness.

Table 1 compares the best validation accuracies and the corresponding summed squared dissimilarities. The hyperspherical mixed prototype approach outperforms the best-performing baseline by 2.29 (the dissimilarity objective) and 1.89 (the MCE objective) percentage points.

[1] https://github.com/DimonLuk/hyperspherical_prototype_mixing

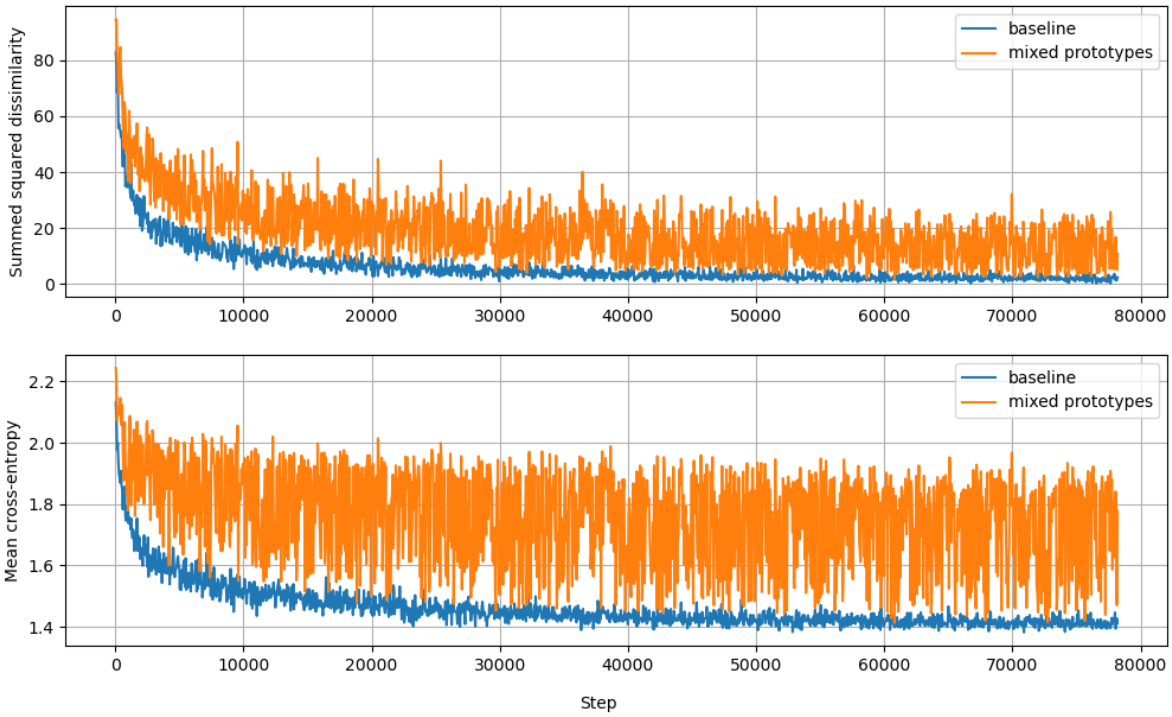

Рис. 1: Comparison of train losses between the baseline and the proposed approaches.

This improvement is significant, considering the baseline accuracies are close to 0.9. Summed squared distance is improved by 35.54% (the dissimilarity objective) and 22.84% (the MCE objective) with respect to the best baseline dissimilarity value.

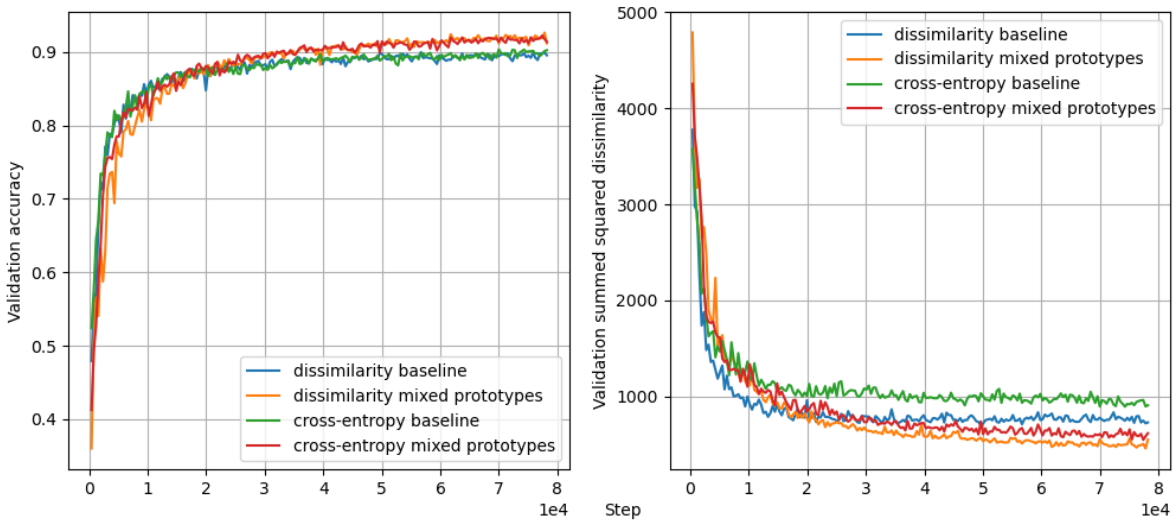

Рис. 2: Comparison of the validation accuracies between the baseline and the proposed approaches.

	Dissimilarity baseline	MCE baseline	Dissimilarity with mixed prototypes	MCE with mixed prototypes
Validation accuracy	0.8998	0.9035	0.9264	0.9224
Summed squared dissimilarity	709	893	457	547

Табл. 1: The best validation accuracies and the corresponding summed squared dissimilarities for the two objectives and the two approaches

4.2 CIFAR 100

For CIFAR 100, The images are upscaled from 32x32 to 64x64, and the ResNet18 model is used. Embedding dimensions d are set to 512.

Training losses are shown in Figure 3. In the case of CIFAR 100, the discrepancy between losses without mixing and with it is even greater than with the CIFAR 10 dataset. The same can be said about the discrepancy between the accuracies of the two approaches. As Figure 4 shows, the accuracy of hyperspherical mixed prototype networks is much better than that of conventional HPNs. It is also worth noting that the difference between the dissimilarity and MCE objectives becomes more apparent when mixed prototypes are used. The dissimilarity loss consistently outperforms the MCE loss in terms of validation accuracy and the clustering of embeddings around their prototypes.

Table 2 compares the best validation accuracies and the corresponding summed squared dissimilarities. The difference between the best baseline validation accuracy and the dissimi-

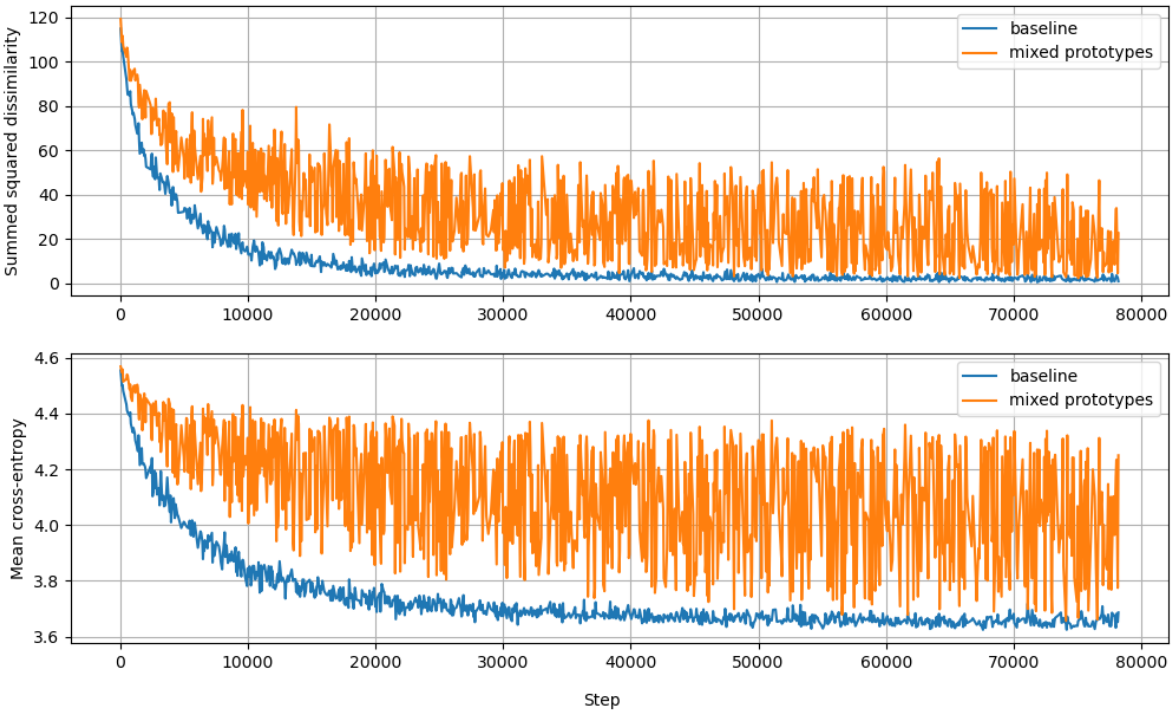

Рис. 3: Comparison of train losses between the baseline and the proposed approaches.

larity objective is 8.1 percentage points, and between the MCE objective is 6.92 percentage points. These results show that the proposed approach provides a significant improvement in terms of accuracy. The difference in summed squared dissimilarity is, respectively, 24.83% and 10.5%.

Another interesting finding is that when mixing is applied, there appears to be a notable difference in terms of the summed squared dissimilarity between the two objective functions and the accuracy. This difference is 1.18 percentage points and may not seem important, but as Figure 4 shows, this phenomenon is consistent. So, there is a hint that dissimilarity minimization can be a better objective for HMPNs.

It is also worth noting that while using a smaller ResNet18 network and randomly initialized prototypes optimized to be as far away from each other as possible, the results for CIFAR 100 presented in this work have a better validation accuracy than the best results reported in the original hyperspherical prototype networks [9] where the authors use larger ResNet34 and privileged information to set up the embedding space. In that work, the reported validation accuracy is 0.65. This finding presents a question: Can combining HMPNs and privileged information provide further performance gains?

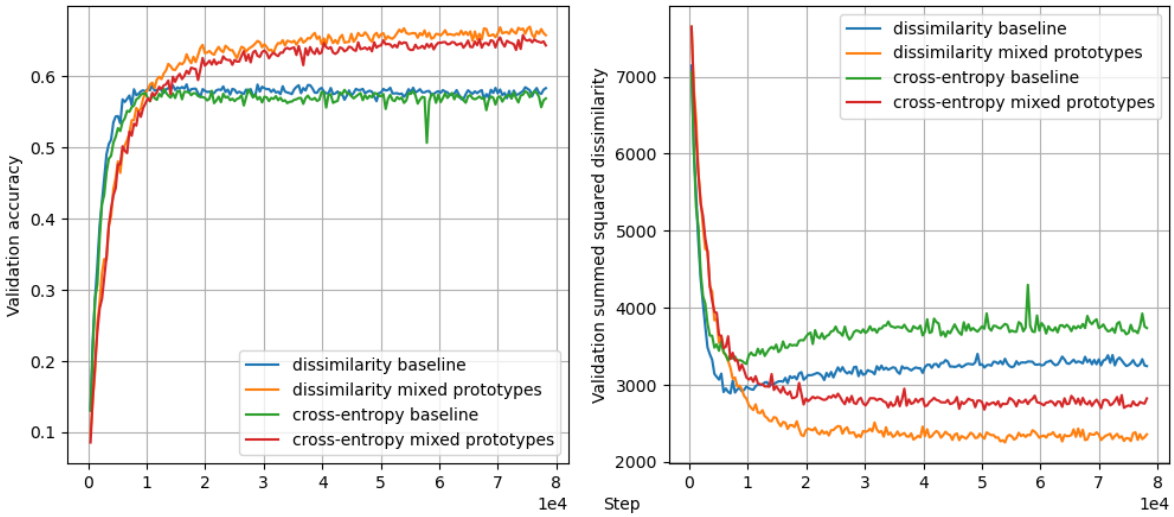

Рис. 4: Comparison of the validation accuracies between the baseline and the proposed approaches.

	Dissimilarity baseline	MCE baseline	Dissimilarity with mixed prototypes	MCE with mixed prototypes
Validation accuracy	0.5886	0.5825	0.6696	0.6578
Summed squared dissimilarity	3022	3376	2271	2703

Табл. 2: The best validation accuracies and the corresponding summed squared dissimilarities for the two objectives and the two approaches

4.3 Tiny ImageNet

For this dataset, the images are kept at their native resolution of 64x64, and ResNet34 is used as a model. Embedding dimensions d are set to 512.

Training losses are shown in Figure 5. The dynamic observed with the previous datasets repeats here. That is, the volatility of the losses increases while validation accuracy and summed squared dissimilarity improve. This effect is shown in Figure 6. In this experiment, the difference between the accuracies of the two objective functions with mixing is even more pronounced than before. The same is true about the summed squared dissimilarities.

Table 3 presents the differences between the best-performing checkpoints in more detail. There are several interesting things to point out. First, as usual, the difference in accuracy between the best baseline and the two objective functions is 7.74 percentage points for the dissimilarity objective and 5.34 for the MCE objective. For summed squared dissimilarities, the difference is, respectively, 12.25% and -0.55%. The minus sign means the baseline clustered

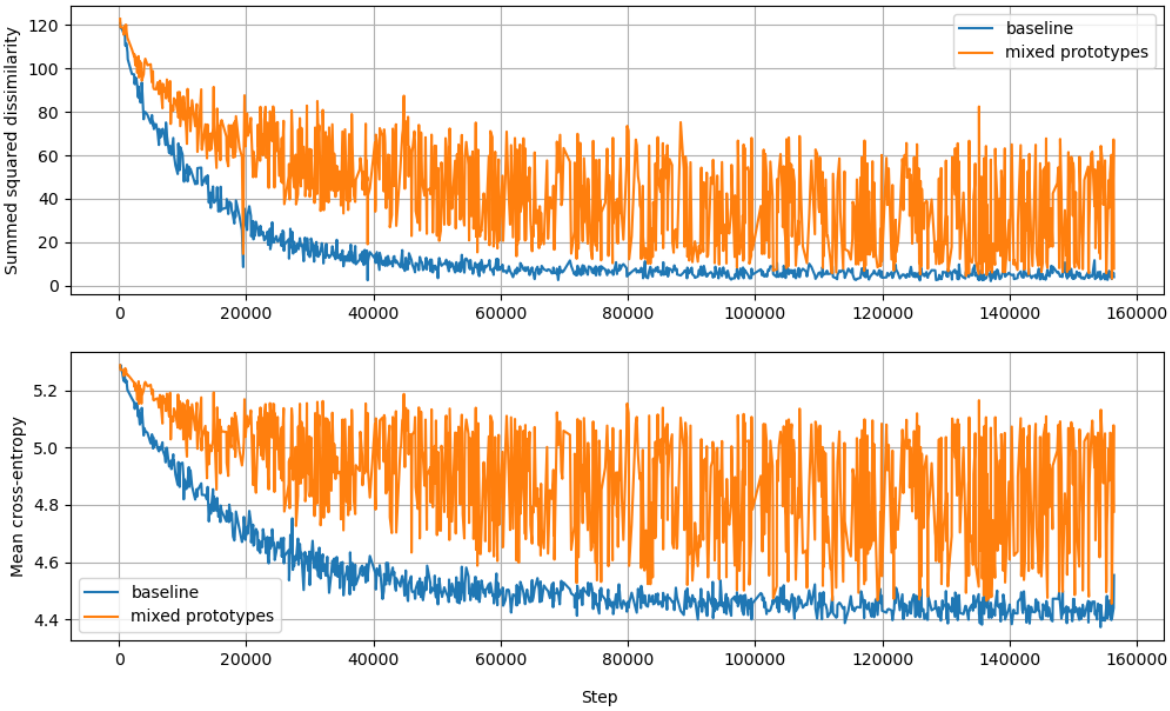

Рис. 5: Comparison of train losses between the baseline and the proposed approaches.

the embeddings better around their corresponding prototypes than the MCE objective with mixed prototypes. This is an interesting finding because the accuracy of the proposed approach remains better even though the clustering is worse. The difference between the two objectives with mixed prototypes is also larger, under these conditions it is 2.24 percentage points in accuracy and 12.72% in summed squared dissimilarities. Moreover, such stark differences are not one-off outliers. They are consistent through the iterations. Finally, the MHPNs' results for Tiny ImageNet outperform the results presented in the HPNs paper by 6.37 percentage points.

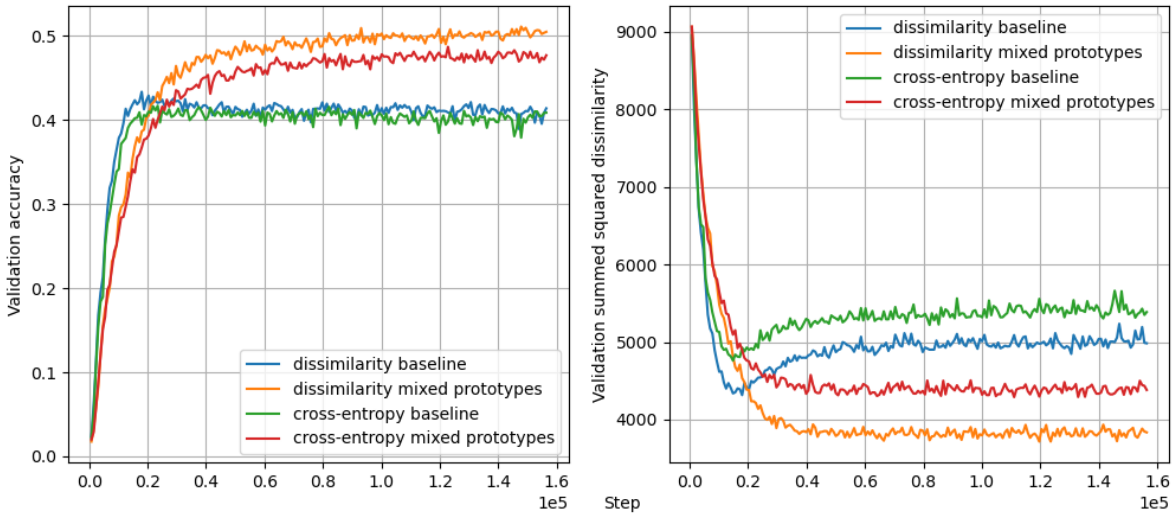

Рис. 6: Comparison of the validation accuracies between the baseline and the proposed approaches.

	Dissimilarity baseline	MCE baseline	Dissimilarity with mixed prototypes	MCE with mixed prototypes
Validation accuracy	0.4333	0.4167	0.5107	0.4867
Summed squared dissimilarity	4315	4924	3786	4338

Табл. 3: The best validation accuracies and the corresponding summed squared dissimilarities for the two objectives and the two approaches

5 Discussion

The experiments show that the hyperspherical mixed prototype approach consistently outperforms the baseline hyperspherical prototype approach across different datasets, task complexities, and model sizes. It is also clear that the hyperspherical mixed prototypes approach is not yet ready to compete with more classical SOTA classifiers, however, the method proposed in this paper makes one step towards that goal. The experiments also demonstrate that the dissimilarity minimization objective results in better performance and intraclass clustering around the prototypes than MCE minimization when using mixed prototypes. The relationship between model sizes, embedding dimensions, mixing strategies, and the performance difference between the two objectives remains to be established. The experiments also show that the difference in accuracy is negligible when the baseline approach is used, while the difference in intraclass clustering remains. Better intraclass clustering may imply a higher level of robustness and is essential for the downstream application of networks

(few-shot and zero-shot learning, anomaly detection, etc.).

There are also open questions, such as how performance improvement depends on the number of classes, embedding dimensions, model size, and model architecture. One of the most influential factors is, probably, the number of dimensions of final projection. Another questions include: will combining prototypes with privileged information with the mixed prototype approach result in further performance gains? What are the applications of other, possibly more aggressive, mixing strategies? Is it possible to extend the approach to other modalities? These questions can be directly translated into research topics that build on the idea introduced in this paper.

6 Conclusions

This paper proposes hyperspherical mixed prototype networks. The concept is built on top of the existing hyperspherical prototype networks approach. The key change is that data samples and the corresponding class prototypes are mixed in their respective spaces. The network optimization objective then becomes to minimize the dissimilarity between mixed samples' embeddings and the corresponding mixed class prototypes or to minimize cross-entropy between the dot product of mixed samples' embeddings with the original prototypes and the corresponding vectors representing the class prototypes' mixtures.

Cutmix and mixup are the mixing strategies chosen because they have been proven to increase neural networks' performance on image classification tasks.

The experiments show that the proposed method improves the performance of deep learning models compared to the baseline HPNs approach for both optimization objectives. The results demonstrated in this paper outperform those presented in the original HPN paper (even those using privileged information to set up prototypes). Another key finding is that for HMPNs and partially for HPNs, the cross-entropy loss does not work as well as expected, and dissimilarity minimization turns out to be a better choice for a loss function.

Potential future work for hyperspherical mixed prototype networks naturally involves investigating how performance gains and the difference between the two objectives depend on the number of classes, embedding dimensions, model sizes, space structures, and architectures. It also includes combining it with prototypes that are set up using privileged information, experimenting with different and, potentially stronger, mixing techniques, as well as investigating the impact of the HMPNs approach on the tasks frequently performed by prototype networks, such as few-shot learning, zero-shot learning, anomaly detection, etc. Another branch of research can focus on extending the approach to other modalities, which also involves experimenting with different mixing techniques. It might also be interesting to investigate further the effects of different optimization objectives on the performance of hyperspherical mixed prototype networks.

References

[1] Sangdoo Yun et al. "CutMix: Regularization Strategy to Train Strong Classifiers With Localizable Features". B: *2019 IEEE/CVF International Conference on Computer*

Vision (ICCV) (2019), c. 6022—6031. URL: https://api.semanticscholar.org/CorpusID:152282661.

[2] Hongyi Zhang et al. "mixup: Beyond Empirical Risk Minimization". B: *International Conference on Learning Representations*. 2018. URL: https://openreview.net/forum?id=r1Ddp1-Rb.

[3] Jin-Ha Lee et al. "SmoothMix: a Simple Yet Effective Data Augmentation to Train Robust Classifiers". B: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2020, c. 3264—3274. DOI: 10.1109/CVPRW50498.2020.00386.

[4] Runji Liu et al. "Attentive Mix: An Efficient Data Augmentation Method for Object Detection". B: *2021 7th International Conference on Computer and Communications (ICCC)*. 2021, c. 770—774. DOI: 10.1109/ICCC54389.2021.9674718.

[5] Jang-Hyun Kim, Wonho Choo and Hyun Oh Song. "Puzzle Mix: Exploiting Saliency and Local Statistics for Optimal Mixup". B: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, 13–18 Jul 2020, c. 5275—5285. URL: https://proceedings.mlr.press/v119/kim20b.html.

[6] A. F. M. Shahab Uddin et al. "SaliencyMix: A Saliency Guided Data Augmentation Strategy for Better Regularization". B: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=-M0QkvBGTTq.

[7] Shaoli Huang, Xinchao Wang and Dacheng Tao. "SnapMix: Semantically Proportional Mixing for Augmenting Fine-grained Data". B: *AAAI Conference on Artificial Intelligence*. 2020. URL: https://api.semanticscholar.org/CorpusID:228063878.

[8] JangHyun Kim et al. "Co-Mixup: Saliency Guided Joint Mixup with Supermodular Diversity". B: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=gvxJzw8kW4b.

[9] Pascal Mettes, Elise van der Pol and Cees Snoek. "Hyperspherical Prototype Networks". B: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach et al. Vol. 32. Curran Associates, Inc., 2019. URL: https://proceedings.neurips.cc/paper_files/paper/2019/file/02a32ad2669e6fe298e607fe7cc0e1a0-Paper.pdf.

[10] Tejaswi Kasarla et al. "Maximum class separation as inductive bias in one matrix". B: *Proceedings of the 36th International Conference on Neural Information Processing Systems*. NIPS '22. New Orleans, LA, USA: Curran Associates Inc., 2024. ISBN: 9781713871088.

[11] Mina Ghadimi Atigh, Martin Keller-Ressel and Pascal Mettes. "Hyperbolic Busemann Learning with Ideal Prototypes". B: *Advances in Neural Information Processing Systems*. Ed. by A. Beygelzimer et al. 2021. URL: https://openreview.net/forum?id=c_XcmuxwAY.

[12] Federico Pernici et al. "Regular Polytope Networks". В: *IEEE Transactions on Neural Networks and Learning Systems* 33.9 (2022), с. 4373—4387. DOI: 10.1109/TNNLS.2021.3056762.

[13] Federico Pernici et al. "Maximally Compact and Separated Features with Regular Polytope Networks". В: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. Черв. 2019.

[14] Jiankang Deng et al. "ArcFace: Additive Angular Margin Loss for Deep Face Recognition". В: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, с. 4685—4694. DOI: 10.1109/CVPR.2019.00482.

[15] Y. Huang et al. "CurricularFace: Adaptive Curriculum Learning Loss for Deep Face Recognition". В: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2020), с. 5900—5909. URL: https://api.semanticscholar.org/CorpusID:209050760.

[16] Xinlong Yang et al. "Prototypical Mixing and Retrieval-based Refinement for Label Noise-resistant Image Retrieval". В: *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2023, с. 11205—11215. DOI: 10.1109/ICCV51070.2023.01032.

[17] Jake Snell, Kevin Swersky and Richard S. Zemel. "Prototypical Networks for Few-shot Learning". В: *Neural Information Processing Systems*. 2017. URL: https://api.semanticscholar.org/CorpusID:309759.

[18] Hong-Ming Yang et al. "Convolutional Prototype Network for Open Set Recognition". В: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44.5 (2022), с. 2358—2370. DOI: 10.1109/TPAMI.2020.3045079.

[19] Mengye Ren et al. "Meta-Learning for Semi-Supervised Few-Shot Classification". В: *International Conference on Learning Representations*. 2018. URL: https://openreview.net/forum?id=HJcSzz-CZ.

[20] Zhixiong Zeng et al. "PAN: Prototype-based Adaptive Network for Robust Cross-modal Retrieval". В: *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '21. Virtual Event, Canada: Association for Computing Machinery, 2021, с. 1125—1134. ISBN: 9781450380379. DOI: 10.1145/3404835.3462867. URL: https://doi.org/10.1145/3404835.3462867.

[21] J.J. Thomson. "XXIV. On the structure of the atom: an investigation of the stability and periods of oscillation of a number of corpuscles arranged at equal intervals around the circumference of a circle; with application of the results to the theory of atomic structure". В: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 7.39 (1904), с. 237—265. DOI: 10.1080/14786440409463107. URL: https://doi.org/10.1080/14786440409463107.

[22] A. Krizhevsky and G. Hinton. "Learning multiple layers of features from tiny images". В: *Master's thesis, Department of Computer Science, University of Toronto* (2009).

[23] Ya Le and Xuan S. Yang. "Tiny ImageNet Visual Recognition Challenge". В: (2015). URL: https://api.semanticscholar.org/CorpusID:16664790.

[24] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". В: *CoRR* abs/1412.6980 (2014). URL: https://api.semanticscholar.org/CorpusID:6628106.

[25] Ekin Dogus Cubuk et al. "RandAugment: Practical Automated Data Augmentation with a Reduced Search Space". В: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, с. 18613—18624. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/d85b63ef0ccb114d0a3bb7b7d808028f-Paper.pdf.

[26] Kaiming He et al. "Deep Residual Learning for Image Recognition". В: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Черв. 2016.

Received 10.01.2026

Лукашов Д.С. *Гіперсферичні змішані прототипні мережі* // Буковинський матем. журнал. — 2026. — Т.14, №1. — С. 54–69.

У цій статті представлено гіперсферичні змішані прототипні мережі. Ключова відмінність у порівнянні з гіперсферичними прототипними мережами полягає у змішуванні прототипів класів та уточнених цілях оптимізації. У цій роботі пропонується змішування зразків даних та відповідних прототипів класів у їхніх відповідних просторах. У цьому випадку метою є максимізація косинусної подібності між змішаним зразком та його відповідним змішаним прототипом. Іншою запропонованою метою є мінімізація перехресної ентропії між скалярним добутком змішаного зразка з оригінальними прототипами та відповідним вектором, що представляє частку кожного прототипу в суміші. Експерименти показують покращення продуктивності у завданні візуальної класифікації порівняно з базовими гіперсферичними прототипними мережами для обох цілей оптимізації. Крім того, результати, представлені в цій роботі, перевершують результати, наведені в оригінальній статті про гіперсферичні прототипні мережі. Ще одним ключовим висновком є те, що при використанні змішування максимізація подібності як цілі оптимізації призводить до кращої точності та набагато кращої внутрішньокласової кластеризації вбудовувань навколо їхніх відповідних прототипів, ніж перехресна ентропія.