Feature Embedding by Template Matching as a ResNet Block

Ada Görgün
ada.gorgun@metu.edu.tr
Yeti Z. Gürbüz
yeti@metu.edu.tr
A. Aydın Alatan
alatan@metu.edu.tr

Dept. of Electrical and Electronics Eng.
& Center for Image Analysis (OGAM)
Middle East Technical University
Ankara, Turkey

Abstract

Convolution blocks serve as local feature extractors and are the key to success of the neural networks. To make local semantic feature embedding rather explicit, we reformulate convolution blocks as feature selection according to the best matching kernel. In this manner, we show that typical ResNet blocks indeed perform local feature embedding via template matching once batch normalization (BN) followed by a rectified linear unit (ReLU) is interpreted as arg-max optimizer. Following this perspective, we tailor a residual block that explicitly forces semantically meaningful local feature embedding through using label information. Specifically, we assign a feature vector to each local region according to the classes that the corresponding region matches. We evaluate our method on three popular benchmark datasets with several architectures for image classification and consistently show that our approach substantially improves the performance of the baseline architectures.

1 Introduction

Convolutional neural networks (CNN), especially ResNet-like [10] architectures [13, 21, 26, 28], are state-of-the-art in image recognition until very recently [6]. The success of CNNs heavily relies on hierarchical feature extraction [30] through stacked convolution blocks (*i.e.*, convolution followed by activation functions) whose parameters are learned in *top-down* manner (*i.e.*, via feedback from class-supervised loss function). A possible explanation for the effectiveness of hierarchical feature extraction is considering each pixel in an intermediate feature map as a feature vector corresponding to a

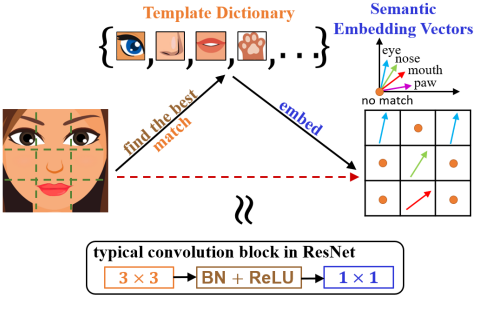

Figure 1: Visualization of our formulation for local feature embedding and its relation to the typical convolution block existing in ResNet.

semantic entity whose existence with other such features forms some other semantic entities in the successors of the hierarchy (*e.g. wing* and *beak* $\rightarrow$ *bird*).

Although this folklore is empirically studied in [29, 30, and references therein] and further verified for attribute-based zero-shot classification in [1, 27], its algorithmic implications for *bottom-up* feature extraction are not clear. Thus, the advances typically focus on architectural designs [13, 21, 26, 28] and leave bottom-up feature embedding formulation rather implicit, which might be a lost opportunity in improving the classification performance. Granted that top-down class-supervised feedback is able to shape the bottom-up behaviour through convolutional layers, can we make hierarchical feature extraction more explicit by exploiting supervision in lower levels?

If we were given localized annotations for lower level features in addition to the class labels, all we need would be a bottom-up feature extraction formulation to exploit such supervision. Nevertheless, we do not have such annotations in practice, which makes explicit supervision of intermediate layers a challenge. That being said, it is shown in text domain [17] that linear combination of the vectors corresponding to semantic entities yields the vector of another entity (*e.g. woman* + *royal* $\approx$ *queen*). Then the question is *"Can we use mixture of class labels to supervise lower level feature extraction?"*

In this study, we address the challenge of using class-level supervision to explicitly shape the behavior of the intermediate features, which differs from building classifiers at the output of intermediate layers to alleviate vanishing gradient as in GoogLeNet [20] like architectures. We first consider bottom-up formulation of feature embedding through template matching and rigorously show its resemblance to how typical ResNet convolution blocks operate (Fig. 1). Building on such a relation, we propose a residual block that assigns a feature vector to each local region according to the classes that the corresponding region matches. We define *best-matching* as a solution of an optimization problem and employ a *soft-max* solution for not only enabling learning but also yielding novel semantic entities as the convex combination of the class features. Specifically, our block is trained with class-level supervision and each local region is encouraged to predict the class of the image it belongs. Surely, some regions are expected to match multiple classes since local features are shared among the classes (*e.g. wing* can exist in both *plane* and *bird*). Our method exploits such information to assign semantically meaningful embedding vectors to those regions by combining the vectors of the matched classes. Namely, we explicitly shape the bottom-up behavior of CNNs by learning to combine existing classes to make up new classes for the local regions. We validate our theoretical claims and show the effectiveness of our method with extensive evaluations on 3 popular classification benchmarks.

2 Related Work

We discuss the works that are most related to ours. Briefly, our contributions include that *i*) we re-formulate ResNet block as a feature embedding by template matching, *ii*) we introduce a batch-statistics-free replacement of BN+ReLU, *iii*) we develop a residual block that effectively combines the embedding vector of the existing classes to yield embedding vectors to different semantic entities.

Related to interpretive feature embedding, bag of visual words based feature aggregation [2] and matching [9] formulations are revisited for global representations.

Those approaches build on feature embedding at the top level of CNN's feature extraction hierarchy. On the contrary, our approach explicitly exploits top-down information in earlier stages of the feature extractors for learning their parameters. That being said, our block employs auxiliary classification loss during training similar to *deeply-supervised nets* [16, 20]. Those methods employ such loss only in training phase to regularize the features and to facilitate learning without vanishing gradients. Differently, we explicitly use predictions in both training and inference to semantically represent local regions with the combination of class specific vectors, which is a novel approach to use auxiliary loss in intermediate layers.

Our work is mostly related to approaches that are inspired from *attention mechanism* [23] of natural language processing to express a token in terms of aggregated features within its context. Interpreting convolution as weighted aggregation of local features, predecessors [12, 18, 24] replace convolution operation entirely with self-attention for bottom-up design of feature extraction. Albeit self-attention is later proven to express any convolutional layer [6], patch-matching based vision transformer (*ViT*) [8] shows no such convolution-mimicking attention layer is essential for powerfully expressive models. In our work, our template matching based formulation is also aligned with attention mechanism. Our work differs in that we arrive at similarity-weighted feature aggregation from formally defining the feature embedding through an optimization problem.

As a byproduct connection, activity normalization methods are related to our technique as well. As the pioneer, batch normalization (BN) [14] addresses *internal covariate shift* phenomenon. Our theoretical results show that BN has an alternative purpose in BN-ReLU context as *pseudo arg-max* optimizer. Such a relation suggests *margin augmented soft-max*[1] as an alternative replacement of BN-ReLU to the existing approaches [3, 22, 25] proposed for the relatively small mini-batches.

3 Method

We repurpose *residual blocks* of a typical residual network [10] as *feature embedding by template matching* and accordingly, propose a novel residual block (depicted in Fig. 2) that effectively learns local feature embedding from class labels.

We first re-formulate convolution block based local feature embedding as feature assignment through best matching kernel. Relating BN-ReLU to arg-max optimizer, we show that the convolution block of 3x3-BN-ReLU-1x1 inherently performs local feature embedding via selecting the best matching convolution kernel (Fig. 1). Hence, inspiring from feature embedding by kernel matching interpretation, we develop our residual block.

3.1 Feature Embedding by Template Matching

We are given a feature map, $f \in \mathbb{R}^{w \times h \times d}$, which is the output of some NN layer. At each spatial location (*i.e.*, pixel), we have a feature $x \in \mathbb{R}^d$ that possibly represents a local region around it to some spatial extent.

We want to obtain a feature map, $f' \in \mathbb{R}^{w' \times h' \times d'}$, from f by transforming x into another vector that captures the semantics of local neighborhood. We let $x_{3\times3} \in \mathbb{R}^{9d}$

[1] A constant is concatenated to the input vector of soft-max.

denote concatenated features of 3x3 window centered at x. We have a set of matching kernels $\{\omega_k \in \mathbb{R}^{9d}\}_k$ each of which seeks for a particular pattern. To each kernel ω_k, we associate an embedding vector, $\nu_k \in \mathbb{R}^{d'}$, representing the semantics of the corresponding 3x3 pattern. We aim to replace x with the embedding vector of the best matching kernel to its neighborhood. Hence, we formally define the problem as:

$$p^* = \underset{\substack{p,q\geqslant 0 \\ q+\Sigma_k p_k=1}}{\arg\max}\ q\,\mu + \textstyle\sum_k p_k\,\omega_k^\intercal x_{3\text{x}3} \tag{P1}$$

where μ is a threshold to zero out the embedding vector when no kernel is matched with at least μ similarity. p^* is either one-hot or zero vector owing to *total unimodularity* [11] of the constraints. We have $p^* = 0$ when any of the activations, $a_k = \omega_k^\intercal x_{3\text{x}3}$, are no greater than μ. Then, we obtain the representation of x as $x' = \Sigma_k p_k^* \nu_k$.

Given the initial feature map, f, the transformed feature map, f', can be efficiently obtained by 3x3 convolution with kernels $\{\omega_k\}_k$, solving a linear program and 1x1 convolution with vectors $\{\nu_k\}_k$, sequentially. Although computationally efficient, one critical problem with such formulation is that the linear program breaks the back-propagation of the computational graph. Namely, p^* as a function of a is not smooth where $a_k = \omega_k^\intercal x_{3\text{x}3}$.

To alleviate non-differentiability of the linear program, we can use stochastically perturbed optimizers [9]:

$$p^* = \mathbb{E}_{z',z\sim\mathcal{N}(0,I)}\left[\underset{\substack{p,q\geqslant 0 \\ q+\Sigma_k p_k=1}}{\arg\max}\ q\,(\mu+\tfrac{1}{\epsilon}z') + p^\intercal(a+\tfrac{1}{\epsilon}z)\right] \tag{P2}$$

or we can use entropy regularization to make the problem strictly concave and smooth:

$$p^* = \underset{\substack{p,q\geqslant 0 \\ q+\Sigma_k p_k=1}}{\arg\max}\ q\,\mu + p^\intercal a - \tfrac{1}{\epsilon}(q\log q + p^\intercal \log p) \tag{P3}$$

where ϵ in both problems controls how smooth the solution p^* is to be. We will introduce two propositions that ensure the existence of the *Jacobian* $[\frac{\partial p^*}{\partial a}]_{ij} \coloneqq \frac{\partial p_j^*}{\partial a_i}$.

Proposition 3.1 (follows from Lemma 1.5 [11]) *Given samples z', z from standard normal distribution, let $\tilde{p}(z', z) \coloneqq \underset{\substack{p,q\geqslant 0 \\ q+\Sigma_k p_k=1}}{\arg\max}\ q\,(\mu+\frac{1}{\epsilon}z') + p^\intercal(a+\frac{1}{\epsilon}z)$. If p^* is the solution of the problem (P2), then we have:*

$$\frac{\partial p^*}{\partial a} = \mathbb{E}_{z',z\sim\mathcal{N}(0,I)}\left[\epsilon\tilde{p}(z',z)\,z^\intercal\right]$$

Proposition 3.2 *The solution of the problem (P3) admits closed form expression as $p_k^* = \frac{\exp(\epsilon a_k)}{\exp(\epsilon\mu)+\Sigma_{k'}\exp(\epsilon a_{k'})}$ (i.e., soft-max) and we have $\frac{\partial p^*}{\partial a} = \epsilon(\Lambda(p^*) - p^* p^{*\intercal})$ where $\Lambda(p^*)$ is the diagonal matrix with p^* as the diagonal.*

Proof: The results follow from the first order optimality conditions owing to strict concavity. ■

The two propositions enable us to implement the best matching kernel selection as a differentiable layer using soft maximizers. p^* will no longer be a one-hot or zero vector. Granted, the entities of p^* will decay to zero if no activation is greater than μ and we will possibly have multiple non-zero entities otherwise due to soft-max operation. To this end, BN-ReLU can be interpreted as a soft approximation of the problem (P1) as we will show shortly.

3.2 BN-ReLU as a Soft Maximizer of (P1)

BN [14] and its successor counterparts [6, 22, 25] perform activity normalization of the form $\hat{a}_k = \gamma_k \frac{a_k - \mathbb{E}[a_k]}{\sqrt{\mathrm{Var}(a_k)}} + \beta_k$ using some batch statistics. Applying ReLU to $\hat{a}$, we obtain $\hat{p} = \max(\hat{a}, 0)$. Given $\{\nu_k\}_k$ embedding vectors, we compute the output feature as $x' = \Sigma_k \hat{p}_k \nu_k$. Denoting $\eta := \Sigma_k \hat{p}_k$ and $\hat{p}^*_k = \hat{p}_k/\eta$, we can equivalently write $x' = \eta \Sigma_k \hat{p}^*_k \nu_k$, where $\hat{p}^*$ is a feasible solution for problem (P1) and indeed is the optimal solution when all the activations are less than μ_k for $\mu_k = \mathbb{E}[a_k] - \frac{\beta_k \sqrt{\mathrm{Var}(a_k)}}{\gamma_k}$. Moreover, $\hat{p}^*$ preserves the relative ordering of the values in the solution of the problem (P3). In fact, BN maps activations around 0 where we have $e^x \approx 1 + x$, meaning that BN-ReLU is a biased first order approximation for unnormalized soft-max for the non-negative activations. Hence, BN-ReLU can be interpreted as yielding a scaled soft maximizer to the problem (P1).

We support our claims on such a relation with empirical studies (§ 4.1) where we replace BN-ReLU with perturbed maximizer [1] and soft-max layers and scale the output with a constant. Such replacement of BN-ReLU mitigates *batch-statistics* demand in activity normalization.

Showing the approximate equivalence between BN-ReLU and arg-max, we can use convolution block of 3x3-BN-ReLU-1x1 to implement our local feature embedding by template matching. In fact, 3x3-BN-ReLU-1x1 is a typical block exploited in ResNet based architectures [10, 13, 26]. Thus, our formulation of local feature embedding provides a different insight towards explanation of how ResNets succeed. Besides, our formulation suggests that 3x3-BN-ReLU-1x1 convolution block is mimicking *cross-attention* [23] between 3x3 patches and convolution kernels. Namely, 3x3 patches are *queries* and the convolution kernels are the *keys*. Each patch is represented by a vector which is the convex combination of *value* vectors corresponding to *keys*.

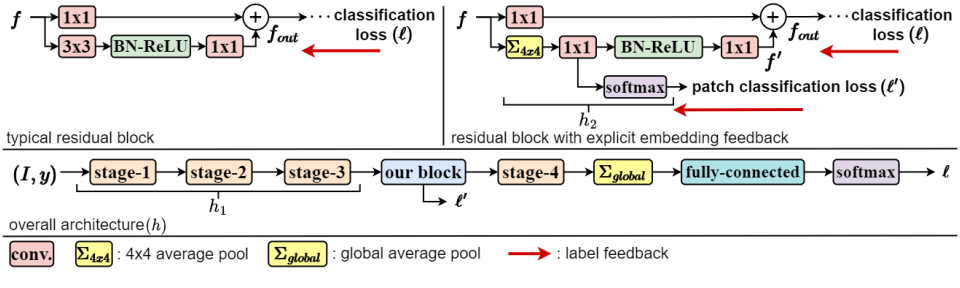

Figure 2: Computation flow of a residual block (top-left), our feature embedding block (top-right), and the overall architecture equipped with our method (bottom).

3.3 Explicit Feature Embedding as Residual Block

We just show that bottom-up behavior of CNNs having 3x3-BN-ReLU-1x1 blocks within is feature vector assignment by template matching. In particular, the embedding vector of a 3x3 patch is the scaled convex combination of the *value* vectors corresponding to the convolution kernels, where combination weights are proportional to the matching scores. Thus, each residual block of ResNet (Fig. 2) can be interpreted as enhancing the feature vectors in the input feature map through shortcut connection by the semantic vectors of the best matching patterns to the corresponding features' 3x3 neighborhood. Following this perspective, we now formulate our feature embedding mechanism.

Instead of 3x3 windows, we consider a larger spatial extent (*i.e.*, *patches*) centered around each pixel in a feature map. Our aim is to match such patches to classes rather than convolution kernels. We achieve this by training an auxiliary classifier for the patches along with the main classifier. Inevitably, the patches having shared entities among classes will not be discriminative enough and will match to multiple classes to minimize the classification loss. We rigorously make use of such behaviour to embed patches regarding their semantic meaning by using learnable embedding vectors, *i.e.*, *value* vectors, for the classes. Specifically, we use the prediction scores to compute convex combination of the *value* vectors. Provided that the learned *value* vectors correspond to semantics of the classes, then their combination will correspond to new semantic entities (*e.g.* $0.5plane + 0.5bird \approx wing$). In this way, we manage to exploit weighted combination of the labels to explicitly supervise local feature extraction.

Formally, given an input feature map, $f \in \mathbb{R}^{w \times h \times d}$, we extract $\frac{w}{2} \times \frac{h}{2}$ patches, $x_\square \in \mathbb{R}^{\frac{w}{2} \times \frac{h}{2} \times d}$ where x with box $\square$ is a patch centered at x. We then obtain a global representation by average pooling for each patch as $x_g = \frac{1}{|x_\square|} \Sigma_{x \in x_\square} x$ where $|x_\square|$ denotes the number of features. We apply a 1x1 convolution (*i.e.*, linear transform) with bias to obtain class matching scores (*i.e.*, activations, a, in the context of our original formulation in § 3.1) for c-many classes as $a_k = \alpha_k^\intercal x_g + \beta_k$ for $k \in [1 \dots c]$ where α_k and β_k are the trainable vector and the bias term for class k.

To learn the classifier parameters, (α, β), we augment the training loss with an auxiliary per patch classification loss. Hence, we are able to propagate label supervision in different levels to explicitly encourage feature embedding by template matching paradigm. The loss for a dataset, $\mathcal{D}$, of image(I)-label(y) tuples becomes:

$$\mathcal{L}(\mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{(I,y) \in \mathcal{D}} \Big[(1\text{-}\lambda) \ell(h(I), y) + \frac{1}{wh} \sum_{x \in h_1(I)} \lambda \ell(h_2(x_\square), y) \Big] \tag{3.1}$$

where $h_1(\cdot)$ denotes the network output of size $w \times h$ until our layer, $h_2(\cdot)$ denotes our layer's class scores, $h(\cdot)$ denotes the whole network's class scores and $\ell(\cdot)$ is the *cross-entropy* loss of soft-maxed scores.

Finally, following our results from §§ 3.1 and 3.2, we apply BN-ReLU-1x1 convolution block to obtain the final representation, $x' \in \mathbb{R}^d$, for the patch $x_\square$. Namely, to each class, we associate an embedding vector, $\nu_k \in \mathbb{R}^d$, to describe the whole patch as $x' = \Sigma_k \hat{p}_k \nu_k$ where $\hat{p}$ is the output of BN-ReLU as we explain in § 3.2. We should note that we use soft-max in loss computation to have normalized probabilities and we rigorously use BN-ReLU for the mixing coefficients to tackle no-match cases while soft-maxing. Hence, our method matches local regions to the class labels rather than

particular patterns and embeds the corresponding semantic information as the scaled convex combination of the class semantics so that the embedded semantic is to be useful in the further levels of the feature embedding hierarchy. Similar to typical residual block, we add the resultant feature map, f', to the initial map, f, via shortcut connection with a per-pixel linear transform, *i.e.*, $f^{out} = \text{conv}_{1\times1}(f) + f'$.

3.4 Implementation Details

We use ResNet (RN) [10], Wide-ResNet (WRN) [28] of *depth* 16 and *widening factor* 2, and DenseNet (DN) [13] of *depth* 100 and *growth rate* 12 as the baseline architectures each of which has 4 stages. In RN and WRN, we have spatial reduction in stage-2 and stage-3 whereas in DN, we have spatial reduction in the first two stages. We summarize the general architecture in Fig. 2 where we also show our feature embedding mechanism as well as $h_1(\cdot)$ and $h_2(\cdot)$ in Eq. (3.1). We place our layer in between the last two stages. We only add an extra classification and two linear transforms (*i.e.*, three 1x1 convolutions) to the baselines. For DN, we additionally employ concatenation of f' and f instead of addition through shortcut to align with the architectural design of DN. We provide further details for reproducibility in the supplementary material.

4 Experimental Work

We evaluate the effectiveness of the proposed feature embedding method for the image recognition task. We further perform ablation studies for the implications of our formulations as well as the effects of the hyperparameters.

Datasets. 100-class Mini-ImageNet [19] with images of size 84x84 and Cifar [15] (10 and 100) with images of size 32x32. We use splits of 65%, 15%, 20% for train, validation, test sets with train data augmentation of [10].

Training. Default *Adam* optimizer with 10^{-3} learning rate, 10^{-4} weight decay, and mini-batch size of 32.

Hyperparameters. We set λ=0.5 in Eq. (3.1) based on our ablation study (Fig. 4). Due to larger images of Mini-ImageNet, we employ additional spatial reduction in the first stages of RN and WRN, and in the third stage of DN to have similar output feature size with Cifar.

4.1 Ablation Studies

Replacing BN-ReLU with soft-maximizers. To support our claims in § 3.2, we replace BN-ReLU following 3x3 convolution with perturbed maximizer [1] and soft-max layers with μ and η constants from § 3.2. In particular, we concatenate μ to activations and perform soft-max, which we refer *margin augmented soft-max*. We then scale the output by η. Using $\mu_k = \mathbb{E}[a_k] - \frac{\beta_k \sqrt{\text{Var}(a_k)}}{\gamma_k}$, we estimate μ=2.5 from BN layers of a pre-trained ResNet20 as the average of non-zero μ_k for each activation. Similarly, we use η=17 from the average of per-pixel sum of the activations after BN-ReLU.

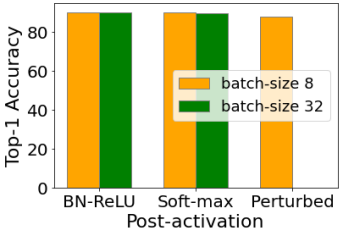

Figure 3: Replacing BN-ReLU

For perturbed maximizer [4], we use 600 samples for empirical expectation. We use ϵ=1 for both methods based on the ablation study in [4]. We evaluated the methods with relatively small (8) and larger (32) batch sizes except we exclude perturbed maximizer in 32 batch size due to its memory demand. We use 3-stage 2-block ResNet20 [10] baseline and Cifar-10 dataset in our evaluation. The comparisons are provided in Fig. 3. We observe that the methods perform on par with each other. Supporting our claims in § 3.2, such empirical results also suggest a technique for activity normalization without using batch-statistics.

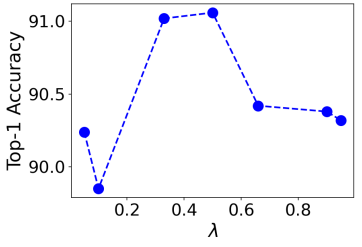

Figure 4: Effect of λ.

Effect of λ. We perform grid search on λ mixing coefficient for the two losses in Eq. (3.1) (*i.e.*, ℓ and ℓ' in Fig. 2). We use 4-stage 2-block ResNet with our method and Cifar-10 dataset in our evaluation. We provide the results in Fig. 4. Small λ values (*i.e.*, absence of auxiliary loss) degrades the performance. We find that equally weighting the losses (λ=0.5) brings the best performance.

Number of blocks (*depth*). We evaluate both 2-block (RN26) and 3-block (RN38) stages in RN baseline to examine the effect of our feature embedding with the increased depth. The comparisons are provided in Table 1 where we observe that increased depth boosts the performance of our method. Notably, we also observe that our method with less depth performs on par with the baseline of more depth.

4.2 Classification Results

We train several architectures (RN#, WRN16, DN100) equipped with our feature embedding block (*Baseline*-Ours). The baselines are of different architectural choices with varying depths. Our aim is rather to show the effectiveness of our theoretical derivations than to push state-of-the-art (SOTA) by architecture design. We firmly believe that our experiments are sufficient to validate the effectiveness and the generalization capability of our method as well as our claims.

Table 1: Evaluation on image recognition task. Bold: best in its category. C : the number of classes.

Dataset →		Cifar10	Cifar100	Mini-ImageNet
Architecture ↓	Params	top-1 acc.	top-1 acc.	top-1 acc.
RN26	0.96M+257C	89.52	65.94	60.43
RN26-aux.	0.96M+386C	90.57	66.21	60.70
RN26-Ours	0.98M+516C	**91.06**	**66.78**	**61.23**
RN38	1.42M+257C	90.78	68.15	60.72
RN38-Ours	1.44M+516C	**91.36**	**69.01**	**63.83**
WRN16	1.28M+129C	90.52	67.11	60.73
WRN16-Ours	1.30M+388C	**91.10**	**67.36**	**62.92**
DN100	1.20M+535C	92.62	71.65	65.03
DN100-Ours	1.32M+1222C	**92.92**	71.25	68.86
DN100-Ours-C	1.36M+1264C	92.71	**72.14**	**68.93**

In order to minimize the confounding of the factors other than our proposed method, we keep the comparisons as fair as possible following the same experimental settings disclosed in § 4 for all architectures. We provide the results in Table 1 where we mark all results that outperform its baseline counterpart. We observe that we improve the performance of WRN and DN, which are SOTA CNN architectures. Moreover, our method consistently improves all the baselines and predominantly, such improvement does not come from the marginal parameter increase that our method brings. 2-block RN26 with our method is mostly superior to its 3-block

baseline (RN38). In relatively shallow architectures, our method's improvement is more significant. With DN architecture, we also experiment enhancing the features by concatenation (DN-Ours-C) instead of addition (§ 3.4). Concatenation is marginally superior to addition in DN owing to better alignment with the architecture of DN.

We also evaluate RN26 with auxiliary classification loss only as in [10, 20] to show the efficiency of our contribution which is exploiting matching scores as the mixing coefficients for the class embedding vectors. Our method brings consistent improvements in all datasets with respect to direct application of auxiliary classification loss in the intermediate layers.

4.3 Analysis of Feature Embedding Behaviour

We further analyze the effect of our feature embedding mechanism with RN26 in Cifar10 dataset through t-SNE plots of the features (Figs. 5 and 6) as well as sample patches (Fig. 7) corresponding to spatial extent of the features. We sample 80 images for each class and project the pixels at the feature maps to 2D space.

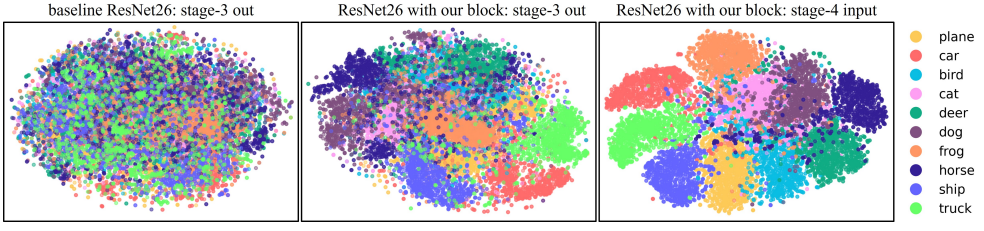

Figure 5: 2D t-SNE projections of the features with and without our method.

Embedding space geometry. We first compare the geometry of the features just before the last stage. We provide the relevant 2D t-SNE projections in Fig. 5. We observe that baseline RN's features are scattered across the space regardless of their higher level semantics. On the contrary, the features at the output of our block (*i.e.*, *stage-4 input*) are clustered with respect to their semantics. In particular, *animals* occupy the one half of the space whereas *vehicles* lie in the other half. We further show that such behaviour is the result of *value* vector embeddings. When we compare the features at the input and the output of our block (*i.e.*, *stage-3 out* and *stage-4 input*), we see that clustering occurs after our feature embedding, validating our mechanism of feature embedding by the matched semantics. That said, in Fig. 6, we also plot the patches according to 2D t-SNE of their class predictions and the resultant embedding vectors as the weighted combination of the class *value* vectors (f' in Fig. 2). With class predictions, semantically similar patches are embedded apart (*e.g. car* and *truck*). On the other

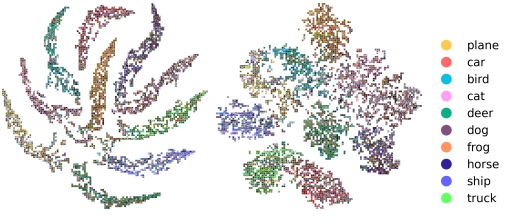

Figure 6: Patches embedded by 2D t-SNE with respect to their class predictions (left) and their embedding vectors (right). Magnified version is available in supplementary.

hand, embedding vectors reshape the geometry so that semantically similar entities are mapped close, yet another result supporting the effectiveness of *feature embedding by template matching* mechanism.

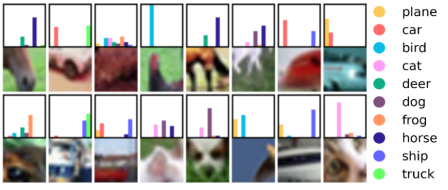

Figure 7: Sample patches with their prediction scores stitched on the top.

Visual words. To support our claims on generating vectors corresponding to new semantic entities from the combination of class vectors, we perform *k-means* clustering with 100 centers of the class prediction scores. We then take the patches that are nearest to the centers. We provide 16 such patches in Fig. 7 together with their prediction scores. We observe that different combination of the classes means different semantic entities. For instance, *wing* is generated by *plane* and *bird* classes, we have *tire* as the combination of *car* and *truck*. We observe class-discriminative patches inheriting the class label. We as well observe more generic entities as the mixture of many classes such as *fur* from *animal* classes.

5 Conclusion

We reformulated convolution block based local feature embedding as feature assignment through best matching kernel and showed that 3x3-soft-max-1x1 implements such a mechanism. Approximately relating BN-ReLU to unnormalized soft-max, we brought a novel view point to 3x3-BN-ReLU-1x1 which we encounter in popular ResNet-based models. Building on perspective explaining the bottom-up behavior of 3x3-BN-ReLU-1x1 convolution block, we proposed a feature extraction mechanism that exploits weighted combination of class-semantic vectors to embed vector representation to the patches. We implemented such mechanism as a simple, yet effective residual layer. Our layer is learnable and effectively selects the classes that matches the patches most for feature embedding. We implemented our method with several architectures. With extensive empirical studies, we validated the effectiveness of our feature embedding layer as well as our theoretical claims.

References

[1] Jacob Abernethy, Chansoo Lee, and Ambuj Tewari. Perturbation techniques in online learning and optimization. *Perturbations, Optimization, and Statistics*, page 223, 2016.

[2] Relja Arandjelovic, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. Netvlad: Cnn architecture for weakly supervised place recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5297–5307, 2016.

[3] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[4] Quentin Berthet, Mathieu Blondel, Olivier Teboul, Marco Cuturi, Jean-Philippe Vert, and Francis Bach. Learning with differentiable pertubed optimizers. *Advances in neural information processing systems*, 33:9508–9519, 2020.

[5] Xiangning Chen, Cho-Jui Hsieh, and Boqing Gong. When vision transformers outperform resnets without pre-training or strong data augmentations. *arXiv preprint arXiv:2106.01548*, 2021.

[6] Jean-Baptiste Cordonnier, Andreas Loukas, and Martin Jaggi. On the relationship between self-attention and convolutional layers. In *International Conference on Learning Representations*, 2019.

[7] Berkan Demirel, Ramazan Gokberk Cinbis, and Nazli Ikizler-Cinbis. Attributes2classname: A discriminative model for attribute-based unsupervised zero-shot learning. In *Proceedings of the IEEE international conference on computer vision*, pages 1232–1241, 2017.

[8] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2020.

[9] Yeti Z. Gürbüz and A. Aydın Alatan. A novel bovw mimicking end-to-end trainable cnn classification framework using optimal transport theory. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 3053–3057, 2019. doi: 10.1109/ICIP.2019.8803276.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, 2016.

[11] A Hoffman, J Kruskal, and M Jünger. Introduction to integral boundary points of convex polyhedra. *Jünger M et al (eds)*, 50:1958–2008, 2010.

[12] Han Hu, Zheng Zhang, Zhenda Xie, and Stephen Lin. Local relation networks for image recognition. In *International Conference on Computer Vision*, 2019.

[13] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *the IEEE conference on computer vision and pattern recognition*, 2017.

[14] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*. PMLR, 2015.

[15] Alex Krizhevsky et al. Learning multiple layers of features from tiny images. 2009.

[16] Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu. Deeply-supervised nets. In *Artificial intelligence and statistics*, pages 562–570. PMLR, 2015.

[17] Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In Yoshua Bengio and Yann LeCun, editors, *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, 2013. URL http://arxiv.org/abs/1301.3781.

[18] Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jon Shlens. Stand-alone self-attention in vision models. *Advances in Neural Information Processing Systems*, 32, 2019.

[19] Sachin Ravi and H. Larochelle. Optimization as a model for few-shot learning. In *ICLR*, 2017.

[20] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[21] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-first AAAI conference on artificial intelligence*, 2017.

[22] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016.

[23] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[24] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE CVPR*, 2018.

[25] Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018.

[26] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[27] Wenjia Xu, Yongqin Xian, Jiuniu Wang, Bernt Schiele, and Zeynep Akata. Attribute prototype network for zero-shot learning. *Advances in Neural Information Processing Systems*, 33:21969–21980, 2020.

[28] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *British Machine Vision Conference 2016*. British Machine Vision Association, 2016.

[29] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[30] Bolei Zhou, David Bau, Aude Oliva, and Antonio Torralba. Interpreting deep visual representations via network dissection. *IEEE transactions on pattern analysis and machine intelligence*, 2018.