Backpropagation-free Contrastive Forward Learning Using Label Embeddings

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

Despite its widespread success, backpropagation is constrained by feedback weight symmetry and forward/backward pass locking. These constraints make backpropagation biologically implausible and computationally inefficient. Free from those problems is recently emerging "forward learning": every layer can update its weights after a forward pass without propagating error signals backward. While forward learning is biologically plausible and computationally efficient, it is hard to formulate a local learning objective in the absence of auxiliary networks. In this work, we propose a simple and biologically plausible way to formulate the local learning objective using label embeddings. Previous works require specialized architecture, input distortion, or local projection, but our method can employ architectures used in backpropagation with minimal modification. Even with small architectures, our method, contrastive forward learning with label embeddings (CFL), outperforms existing forward learning approaches on non-trivial datasets such as CIFAR-10 and CIFAR-100, while approaching performance close to backpropagation. Furthermore, our training objectives allow label embeddings to learn the meaningful representation of the labels, such that embedding interpolation enables zeroshot inference.

1 Introduction

Backpropagation (BP) Rumelhart et al. [1986] has been at the forefront of deep learning. Not only is BP easy to implement through modern deep learning frameworks, but it also offers a simple solution to the credit assignment problem: how each weight of a model should adjust its value to optimize the model's performance Werbos [1974]? Error signals are obtained through the loss function of model outputs and targets. Then, BP allows a model to update all of its weights in a way that reduces the error signals, by backwardpassing the error signals through symmetric feedback weights. Therein lies two constraints for BP: (1) weights used in forward and backward passes should be symmetric and (2) backward passes cannot start until all forward passes are complete, and vice versa (forward and backward locking).

These constraints pose two main problems for BP. Firstly, it is not biologically plausible. If our brain learns by BP, symmetric neural pathways should be prevalent in the brain, but they are not. Biological plausibility is not necessary for a competent learning algorithm. However, much artificial intelligence research still strives to mimic human cognition, since we humans are the most intelligent agents we can find (yet). Such observation suggests that BP may not be an optimal learning algorithm because it is drastically different from how we learn. Secondly, BP is computationally inefficient. Computation of the weight gradients requires local activation to be stored. Accordingly, after a forward pass, each layer consumes memory to store local activation while staying idle, until all backward passes are complete. Such forward/backward locking severely undermines the parallelization of computation and makes BP undesirable for edge computing.

Many backpropagation alternatives have been introduced. Feedback alignment (FA) Lillicrap et al. [2014, 2016] replaces symmetric feedback weights with fixed random feedback weights. The problems of BP persist. FA lifts the weight symmetry constraint, but it is still constrained by reciprocal feedback connection. Moreover, FA still suffers from forward/backward locking. In direct feedback alignment (DFA) Nøkland [2016], fixed random weights backwardpass error signals to each layer directly. DFA is more biologically plausible than FA in a flexible way because DFA does not enforce reciprocal feedback connection, resembling top-down feedback systems in the brain Gilbert and Li [2013]. Yet, although DFA unlocks backward passes, it is still forward-locked: weights cannot be updated until all forward passes are complete. In target propagation (TP) Lee et al. [2014], like BP, each layer learns by reconstructing the activation of the previous layer. Like BP, TP is forward/backward locked.

Using BP only in local modules, local learning (LL) Nøkland and Eidnes [2019], Belilovsky et al. [2019] offers a balanced solution to the problems of BP. With the module-wise weight symmetry, LL achieves improved computational efficiency over BP as it is module-wise forward/backward unlocked. A local module consists of layers used in forward passes as well as auxiliary networks. Only used during training to compute the local loss and propagate error gradients backward, auxiliary networks constitute memory overhead, especially in edge computing. A recent approach "Forward Learning" (FL) Frenkel et al. [2021], Hinton [2022] goes a step further by eliminating module-wise BP and auxiliary networks. In FL, weights of each layer are updated by the layer-wise local loss, thereby making FL completely forward/backward unlocked and especially fit for edge computing. For example, implementing a forward learning method Frenkel et al. [2021], spiking online-learning convolutional neuromorphic processor (SPOON) achieves 16.8% power overhead and 11.8% area overhead on on-chip online learning compared to offline learning Frenkel et al. [2021, 2020]. Despite such advantages, FL has to overcome two main challenges: formulating local targets for loss calculation and poor performance compared to BP and LL. Without auxiliary networks to transform local features, it is hard to use local features and local targets in the local loss calculation because the dimension of the features and the targets do not usually match (e.g. features and labels in classification with CNN). Even if existing methods manage to obtain the local loss without auxiliary networks, their performances are poor on non-MNIST datasets like CIFAR-10. Details are discussed in Section 2.2.

To formulate local label targets, we use label embedding vectors. Just as each word has a corresponding embedding vector in NLP Mikolov et al. [2013], Devlin et al. [2018], each classification label has a corresponding label embedding vector. To formulate local objectives, we leverage contrastive learning. Contrastive learning has been effectively used for representational learning in various settings—supervised, unsupervised, visual, multimodal, etc. Chen et al. [2020a,b], Khosla et al. [2020], van den Oord et al. [2018], Park et al. [2020], Radford et al. [2021]. In this work, we employ two supervised contrastive objectives so that each layer learns a salient representation of inputs without global feedback. One objective maximizes agreement between image features with the same label while minimizing agreement between image features of different labels. The other maximizes agreement between image features and their corresponding label embedding vector while minimizing agreement between image features and their non-matching label embedding vectors.

During training, label embedding vectors learn meaningful label-specific representation such that embedding interpolation enables zeroshot inference. Unlike existing local and forward learning approaches, our method requires no specialized architecture, auxiliary network, input distortion, or local projection, other than a label embedding dictionary. Accordingly, as long as the computational graph is detached after each layer, our method can utilize BP architectures with minimal modification. Extensive experiments demonstrate that our method, contrastive forward learning with label embeddings (CFL), outperforms existing forward learning approaches on the CIFAR-10 and CIFAR-100, even with small architectures.

2 Related Works

Local learning and forward learning are greedy local learning in the sense that each module/layer updates its weights to optimize local features for the local objective Baldi and Sadowski [2015], Belilovsky et al. [2018]. Since inference of most deep learning models only uses the final outputs from the last forward pass, intermediate local features learned from greedy local learning are not optimized for final outputs used in inference. This section discusses how such greedy learning still manages to stay competitive.

2.1 Local Learning

With two separate auxiliary network pathways to generate features for reconstruction and label prediction loss, Nøkland and Eidnes [2019] is one of the first few that outperform BP on CIFAR-10 and CIFAR-100. The reconstruction loss compares L2 distance between self similarity matrix of one-hot encoded labels and local features. The label prediction loss is the cross entropy loss typically used in the final layer in classification. The label prediction loss is the most common local loss in greedy local learning Belilovsky et al. [2018, 2019], Pathak et al. [2022]. In contrast, our method uses no auxiliary network and neither of the local loss functions.

Wang et al. [2021] analyzes greedy local learning in the perspective of information theory. The authors examine how greedy local learning affects $I(x, h)$, mutual information between inputs and local features from auxiliary networks, and $I(h, y)$, mutual information between local features and labels. Compared to BP, $I(x, h)$ and $I(h, y)$ decrease much more quickly the deeper the layer in local learning. The authors prove that minimizing the supervised contrastive loss Khosla et al. [2020] maximizes the lower bound of $I(h, y)$ across layers. Since the contrastive loss function can be used without auxiliary networks, we also adopt it as one of our loss functions (the equation 9).

2.2 Forward Learning

The recently emerging approach, the forward forward algorithm Hinton [2022], overlays one-hot encoded labels onto images to formulate local objective. Inspired by Noise Contrastive Estimation Gutmann and Hyvärinen [2010], FF optimizes local features to be above a certain threshold if the overlaid labels match the images (positive pair). Otherwise (negative pair), local features are optimized to be under a certain threshold. To receive global information, FF also proposes a RNN-variant. The RNN-variant processes the same input for multiple time steps; intermediate layers receive normalized states from the upper layer at the previous time step for global information, similar to top-down processing in the brain. The predictive forward forward algorithms (PFF) Ororbia and Mali [2023] combines the RNN-based FF with predictive coding Rao and Ballard [1999a], Salvatori et al. [2021], Ororbia and Kifer [2020], Rao and Ballard [1999b], in which neurons predict activation of nearby neurons and learn based on the difference between the prediction and observed activation. PFF jointly trains classifier and generative network, such that the generative network learns to reconstruct/predict intermediate features from the classifier and to reconstruct inputs from Gaussian noise. During training, classifier and generative network interacts through lateral and top-down circuits for multiple time steps. PFF performs experiments on only MNIST and its variants, while requiring the generative auxiliary network to train the classifier. Symba Lee and geun Song [2023] improves upon FF by introducing Intrinsic Class Pattern (ICP). Rather than overlaying one-hot encoded label which hides a portion of an input image in FF, Symba adds a label-specific fixed random noise map to a separate image channel. The threshold-based NCE local objectives used in FF-based approaches are unstable and scales poorly on CIFAR-10 and CIFAR-100.

The direct random target projection (DRPT) uses fixed random weights to project one-hot encoded labels into local targets, just as DFA uses fixed random weights to project global error gradients to local layers. Our method doesn't use random weight projection because we directly use label embedding vectors as local targets. In contrast, the cascaded forward algorithm (CAFO) Kirkpatrick et al. [2016] freezes the random weights of feedforward layers and only updates the weights of the local linear projection layers used in local prediction. When the number of labels to predict is small as in MNIST and CIFAR-10, CAFO requires relatively small architecture. However, due to linear projection from a feature map to class predictions, CAFO scales poorly on the number of classes. While projection-based approaches surpass FF-based approaches with smaller architectures, our method outmatches the projection-based approaches on CIFAR-10 and CIFAR-100 with even smaller architectures and without projection layers.

3 Contrastive Forward Learning with Label Embeddings

3.1 Label Embedding

Vectorizing labels into dense vectors, rather than sparse one-hot encoded vectors, our method uses the dense vectors as local targets for the local loss functions. In NLP, each input word has a corresponding embedding vector queried from a vocabulary dictionary Mikolov et al. [2013], Vaswani et al. [2017], Devlin et al. [2018]. During training, the word embeddings receive error gradients from BP. After training, the word embeddings are distributed in the semantically meaningful latent space such that interpolation of embeddings is possible Badimala et al. [2019]. Similarly, we query a label embedding vector corresponding to each label, from a label embedding dictionary. For Z label classes, we have a label embedding dictionary $\boldsymbol{D}^Z = \{\boldsymbol{t}^1, \ldots, \boldsymbol{t}^Z\}$ where $\boldsymbol{t}^z \in \mathbb{R}^{C_D}$. During training, the label embedding vectors and layer weights are optimized to maximize similarity between feature vectors and their corresponding label embedding vector, while minimizing similarity between feature vectors and non-corresponding label embedding vectors (Section 4).

3.2 Training Loss

Consider a model with L layers $F = \{f^1, ..., f^L\}$, each with learnable weights θ^l. Except for the final linear classifier layer f^L, every intermediate layer f^l obtains error gradients from the two local loss functions. Each local layer f^l outputs the local feature map $\mathbf{h}_n^l \in \mathbb{R}^{C_l \times K_l}$ such that $\mathbf{h}_n^l = f^l(sg[\mathbf{h}_n^{l-1}])$, where $sg[\cdot]$ is the stop gradient operator and $\mathbf{h}_n^0$ is the input image, and C_l is the dimension of the feature vectors at the l-th layer. K_l is the number of feature vectors. For a convolutional layer, $K_l = \text{height}_l \times \text{width}_l$, whereas in fully connected layers (FC), layer outputs are divided into K groups ($\mathbb{R}^{CK} \mapsto \mathbb{R}^{C \times K}$). K values are listed in Table 4.

We use the mean local feature vector $\bar{\boldsymbol{h}}_n^l \in \mathbb{R}^{C_l}$, averaged across K, to represent the n-th sample in a batch of N samples. The batch embedding contrastive loss $\mathcal{L}_{\text{batch}}$ takes the mean feature vectors $\bar{\boldsymbol{h}}_n^l$, while the feature contrastive loss $\mathcal{L}_{\text{feat}}$ uses the l2-normalized mean feature vectors $\hat{\boldsymbol{h}}_n^l$. The final prediction layer f^L updates its weights through the cross entropy loss, as in BP training. Prediction is also possible at every layer, by choosing the label with the highest similarity to the feature vectors:

$$\hat{y} = argmax_z(\langle \bar{\boldsymbol{h}}_n^l, \boldsymbol{t}^z \rangle) \text{ for } \boldsymbol{t}^z \in \boldsymbol{D}^Z, \tag{1}$$

where $\langle \cdot, \cdot \rangle$ denotes the dot product. The models without the final linear classifier f^L makes prediction through the equation (1).

3.2.1 Embedding Contrastive Loss

For the batch of N layer outputs $H_N^l = \{\bar{\boldsymbol{h}}_1^l, \ldots, \bar{\boldsymbol{h}}_N^l\}$, we obtain the corresponding batch of label embedding vectors:

$$T_N^l := \{avgpool^l(\boldsymbol{t}_n) + \epsilon \mid y(\boldsymbol{t}_n) = y(\bar{\boldsymbol{h}}_n^l) \text{ for } \boldsymbol{t}_n \in \boldsymbol{D}_l^Z \text{ and } \bar{\boldsymbol{h}}_n^l \in H_N^l,\ \epsilon \sim N(0, \frac{1}{C_l^2})\}, \tag{2}$$

where $avgpool^l : \mathbb{R}^{C_D} \mapsto \mathbb{R}^{C_l}$ is the average pooling with a padding of 0, a stride of $\frac{C_D}{C_l}$, and kernels of size $C_D - (C_l - 1) \times \text{stride}$. During training, label embedding vectors receive error gradients only at the last intermediate layer $L - 1$, such that:

$$\boldsymbol{D}_l^Z = \{sg[\boldsymbol{t}_x] \text{ if } l < L-1,\ \boldsymbol{t}_x \text{ otherwise} \mid \boldsymbol{t}_x \in \boldsymbol{D}^Z\}. \tag{3}$$

Then, the batch embedding contrastive loss for $\bar{\boldsymbol{h}}_n^l$ is:

$$\mathcal{L}_{\text{batch}}^l(\bar{\boldsymbol{h}}_n^l, T_N^l) = -\log \frac{\exp\langle \bar{\boldsymbol{h}}_n^l, \boldsymbol{t}_n \rangle}{\sum_{\boldsymbol{t}_i \in T_N^l, y(\boldsymbol{t}_i) \neq y(\bar{\boldsymbol{h}}_n^l)} \exp\langle \bar{\boldsymbol{h}}_n^l, \boldsymbol{t}_i \rangle}, \tag{4}$$

Negative pairs, such as $\{(\boldsymbol{t}_i, \bar{\boldsymbol{h}}_n^l) \mid y(\boldsymbol{t}_i) \neq y(\bar{\boldsymbol{h}}_n^l),\ \boldsymbol{t}_i \in T_N^l\}$, play a significant role in the performance of contrastive learning Chen et al. [2020a], Khosla et al. [2020]. Since $\boldsymbol{t}^z \in \boldsymbol{D}^Z$ remains constant for each label z, many negative pairs overlap without the noise ϵ in the equation (2). The noise injection ensures 1) each negative pair is unique and 2) $\mathcal{L}_{\text{batch}}$ optimizes similarity between feature vectors and vectors near the label embedding vectors.

We also introduce another embedding contrastive loss. The dictionary contrastive loss replaces T_N^l with $\hat{\boldsymbol{D}}_l^Z$:

$$\hat{\boldsymbol{D}}_l^Z := \{avgpool^l(\boldsymbol{t}_z) + \epsilon \mid \boldsymbol{t}_z \in \boldsymbol{D}_l^Z \text{ and } \epsilon \sim N(0, \frac{1}{C_l^2})\}. \tag{5}$$

Accordingly, the dictionary contrastive loss becomes:

$$\mathcal{L}_{\text{dict}}^l(\bar{\boldsymbol{h}}_n^l, \hat{\boldsymbol{D}}_l^Z) = -\log \frac{\exp\langle \bar{\boldsymbol{h}}_n^l, \boldsymbol{t}^+ \rangle}{\sum_{\boldsymbol{t}_i \in \hat{\boldsymbol{D}}_l^Z} \exp\langle \bar{\boldsymbol{h}}_n^l, \boldsymbol{t}_i \rangle},\ y(\boldsymbol{t}^+) = y(\bar{h}_n^l) \tag{6}$$

$\mathcal{L}_{\text{batch}}$ requires N label embedding vectors, whereas $\mathcal{L}_{\text{dict}}^l$ only needs Z (the number of classes) embedding vectors. As long as Z < N (which is usually the case), $\mathcal{L}_{\text{dict}}^l$ comes with less memory overhead. However, each embedding loss has its advantage and disadvantage. Details are discussed in Section 4.3 and 4.4.

$\mathcal{L}_{\text{batch}}$ and $\mathcal{L}_{\text{dict}}$ are inspired by InfoNCE van den Oord et al. [2018], but we employ the dot product as a similarity measure instead of the cosine similarity, unlike many other works leveraging InfoNCE Radford et al. [2021], Yu et al. [2022], Park et al. [2020]. As the ablation results in the table 6 show, the dot product results in significantly greater performance compared to the cosine similarity. The hypothesis for the results is discussed in Section 4.4.

3.2.2 Feature Contrastive Loss

For the batch of N normalized mean feature vectors $\hat{H}_N^l = \{\hat{\boldsymbol{h}}_1^l, \ldots, \hat{\boldsymbol{h}}_N^l\}$, let:

$$\hat{H}_+^l(z) := \{\hat{\boldsymbol{h}}_n^l \mid z = y(\hat{\boldsymbol{h}}_n^l) \text{ for } \hat{\boldsymbol{h}}_n^l \in \hat{H}_N^l\}, \tag{7}$$

$$\hat{H}_-^l(z) := \hat{H}_N^l - \hat{H}_z^l. \tag{8}$$

$\hat{H}_+^l(z)$ denotes the set of feature vectors positive to the label z; positive feature vectors share the same label z. Likewise, $\hat{H}_-^l(z)$ denotes the set of feature vectors negative to the label z, such that negative feature vectors correspond to any label but z. Then, the feature contrastive loss for the $\hat{\boldsymbol{h}}_n^l$ is:

$$\mathcal{L}_{\text{feat}}^l(\hat{\boldsymbol{h}}_n^l, \hat{H}_N^l) = -\log \left[\frac{1}{|\hat{H}_+^l(z)|} \sum_{\hat{\boldsymbol{h}}_+^l \in \hat{H}_+^l(z)} \frac{\exp\left(\langle \hat{\boldsymbol{h}}_n^l, \hat{\boldsymbol{h}}_+^l \rangle / \tau\right)}{\sum_{\hat{\boldsymbol{h}}_-^l \in H_-^l(z)} \exp\left(\langle \hat{\boldsymbol{h}}_n^l, \hat{\boldsymbol{h}}_-^l \rangle / \tau\right)} \right],\ z = y(\hat{\boldsymbol{h}}_n^l), \tag{9}$$

where τ is a temperature hyperparameter. In our experiments, we use $\tau = 0.07$. Minimizing $\mathcal{L}_{\text{feat}}$ maximizes agreement between feature vectors that belong to the same label and minimizes agreement between feature vectors with different labels.

3.2.3 Total Local Loss

The total local loss function for the l-th layer is as follows:

$$\mathcal{L}_{\text{total}}^l(\{\mathbf{h}_1^l, \ldots, \mathbf{h}_N^l\},\ \boldsymbol{D}^Z) = \frac{1}{N} \sum_{n=1}^{N} (\lambda_1 \mathcal{L}_{\text{batch}}^l(\bar{\boldsymbol{h}}_n^l, T_N^l) + \lambda_2 \mathcal{L}_{\text{dict}}^l(\bar{\boldsymbol{h}}_n^l, \hat{\boldsymbol{D}}_l^Z) + \lambda_3 \mathcal{L}_{\text{feat}}^l(\hat{\boldsymbol{h}}_n^l, \hat{H}_N^l)), \tag{10}$$

where $\lambda_3 = 1$ across all experiments. Training with both embedding losses did not result in improvement over using only $\mathcal{L}_{\text{batch}}$, so we use only one of the embedding losses. Thus, we set $\lambda_1 = 0$ for $\mathcal{L}_{\text{total-D}}^l$ and $\lambda_2 = 0$ for $\mathcal{L}_{\text{total-B}}^l$. The ablation experiments in Section 4.4 highlight that training with $\mathcal{L}_{\text{feat}}$ alone results in significantly poor performances, in contrast to $\mathcal{L}_{\text{batch}}$ and $\mathcal{L}_{\text{dict}}$ which still manage to perform well when used alone. Regardless, $\mathcal{L}_{\text{feat}}$ makes the representation of feature vectors more disentangled and improves performance when used in conjunction with $\mathcal{L}_{\text{batch}}$, as discussed in Section 4.4.

4 Experiments

4.1 Datasets

We test our method on MNIST LeCun and Cortes [2005], CIFAR-10, and CIFAR-100 Krizhevsky [2009] datasets. The MNIST dataset consists of 60000 training samples and 10000 test samples, each of which is a 28×28 grayscale image. The CIFAR-10 and CIFAR-100 datasets each contain 50000 training samples and 10000 testing samples of 32×32 RGB images. There are 100 fine-label classes in CIFAR-100. Every five fine-label classes belong to one coarse-label class; the 100 fine-label classes are grouped into 20 coarse-label classes. For example, fine-label classes $\{maple, oak, palm, pine, willow\}$ constitute the $tree$ coarse-label class. In zeroshot experiments in Section 4.4, we use interpolation of fine-label embedding vectors for zeroshot prediction of the coarse-labels.

4.2 Training Details

Each CFL model is compared with the three baseline methods: BP, FA, and DFA. Each CFL model and its baseline models share the same architecture except for the label embedding dictionary $\boldsymbol{D}^Z$ and the stop gradient operators. Thus, the difference in the number of parameters in Table 1 comes from $\boldsymbol{D}^Z$. The architecture details are in Table 4. Since each layer f^l of the CFL model updates its weights independently of error gradients in other layers, it is possible to fine-tune the learning rate of every layer. For the simplicity of testing and comparison, however, we use the same learning rate for all layers. Moreover, each CFL model and its baseline models are trained using the same training hyperparameters listed in Table 5.

	MNIST		CIFAR-10		CIFAR-100	
	Params	Test err	Params	Test err	Params	Test err
DRTP FC (Frenkel et al. [2021])	1.80M	3.9	4.09M	51.06	19.2M	88.32 *
DRTP CONV (Frenkel et al. [2021])	6.28M	1.33	16.54M	30.59	46.42M	65.02*
FF (Hinton [2022])	1.87M	1.36	18.93M	41	19.2M	99*
CAFO (Zhao et al. [2023]	243K	**1.20**	243K	32.6	2.43M	59.2
PFF (Ororbia and Mali [2023])	23M	1.34	32.37M	50.14*	32.73M	81.3*
Symba (Lee and geun Song [2023])	1M	1.65	16M	41.2		
Symba (Lee and geun Song [2023])	11M	1.42	31M	40.9	31M	70.7
CFL FC ($\mathcal{L}_{\text{total-B}}$)	1.88M	1.69	18.93M	33.97	19.23M	68.4
CFL FC ($\mathcal{L}_{\text{total-D}}$)	1.88M	1.55	18.93M	33.78	19.23M	67.38
CFL conv256 ($\mathcal{L}_{\text{total-B}}$)	153.8K	2.96	155K	24.24	201.1K	51.1
CFL conv256 ($\mathcal{L}_{\text{total-D}}$)	153.8K	2.66	155K	24.56	201.1K	51.4
CFL conv256 ($\mathcal{L}_{\text{total-B}}$ without f^L)	**151.2K**	2.8	**152.4K**	24.77	**175.4K**	51.34
CFL conv256 ($\mathcal{L}_{\text{total-D}}$ without f^L)	**151.2K**	2.74	**152.4K**	27.5	**175.4K**	61.65
CFL conv512 ($\mathcal{L}_{\text{total-B}}$)	1.34M	1.36	1.341M	**16.31**	1.433M	50.43
CFL conv512 ($\mathcal{L}_{\text{total-D}}$)	1.34M	1.25	1.341M	16.49	1.433M	**49.76**
CFL conv512 ($\mathcal{L}_{\text{total-B}}$ without f^L)	1.335M	1.47	1.336M	16.67	1.382M	51.01
CFL conv512 ($\mathcal{L}_{\text{total-D}}$ without f^L)	1.335M	1.34	1.336M	16.84	1.382M	55.84
BP FC	1.871M	1.29	18.93M	35.96	19.21M	67.3
FA FC	1.87M	1.51	18.93M	39.33	19.21M	69
DFA FC	1.87M	1.75	18.93M	45.55	19.21M	86.42
BP conv256	151.2K	2.63	152.4K	22.52	175.4K	48.72
BP conv512	1.335M	1.31	1.336M	14.05	1.382M	46.4
FA conv256	151K	2.85	152K	27.65	175K	52.57
FA conv512	1.33M	1.3	1.34M	19.67	1.38M	52.93
DFA conv256	151K	9.4	152K	45.83	175K	68.56
DFA conv512	1.33M	4.39	1.34M	35.7	1.38M	69.06

Table 1: Classification Accuracy Results. The underlined models are BP baselines trained with the cross entropy loss. The underlined results outscore those of the baselines. The highlighted results indicate best scores among the backprop alternatives. Scores with $*$ are reproduced results.

4.3 Results

We also compare our CFL models against other forward learning models. Table 1 lists the number of parameters for each model and its best test accuracy. For the MNIST dataset, CAFO Zhao et al. [2023] still performs best. However, for more complex datasets—CIFAR-10 and CIFAR-100—our method outshines other forward learning approaches. Our best model, CFL conv512, outperforms other forward learning models by a large margin. Moreover, our smallest model, CFL conv256 (without f^L), still outperforms all other forward learning models although its number of parameters is much smaller than those of other forward learning models. Our fully connected model, CFL FC, also outperforms other fully connected models —DRTP, FF, and Symba. Overall, for each architecture type, our method performs and scales better than other forward learning methods.

CFL performances on MNIST and CIFAR-10 come close to the BP baselines, with both CFL FC models outperforming their BP baseline on CIFAR-10. CFL conv512 ($\mathcal{L}_{\text{total-D}}$) also outperforms its baseline on MNIST. On CIFAR-100, however, the performance gap is larger for the convolutional models. The performances of $\mathcal{L}_{\text{total-B}}$ and $\mathcal{L}_{\text{total-D}}$ are on par with each other when the models are trained with the final classifier f^L. However, without f^L, the models trained with $\mathcal{L}_{\text{total-D}}$ exhibit poor performance compared to those of the models with f^L. Furthermore, for CIFAR-10 and CIFAR-100, our models (except for $\mathcal{L}_{\text{total-D}}$ without f^L) outperform FA and DFA for each architecture.

4.4 Zeroshot and Ablation Experiments

The models trained on CIFAR-100 fine-labels can make zeroshot inferences on CIFAR-100 coarse-labels. By averaging five fine-label embedding vectors belonging to each coarse-label, we obtain 20 coarse-label embedding vectors from 100 fine-label embedding vectors. Then, we can make zeroshot prediction using the equation (1).

The results in Table 2 stress the importance of the noise injection and $\mathcal{L}_{\text{feat}}$, without which the performances worsen across all architectures. This observation is consistent with CIFAR-10 and CIFAR-100 ablation results on Table 6. Figure 1 highlights that zeroshot scores and CIFAR-100 scores are highly correlated. The better the label-specific representation of embedding vectors, the more accurate the embedding interpolation and zeroshot prediction become. Thus, zeroshot performance can serve as a metric for representation learning in our method. For convolutional models, the gap between zeroshot performance and its end-to-end baseline is smaller for the larger conv512 model. This trend is also observed in zeroshot performances of BP models trained with $\mathcal{L}_{\text{total}}^{L-1}$, although the gap between zeroshot performance and its end-to-end baseline is overall smaller for BP. On the other hand, zeroshot performance of our largest model, CFL FC, is on par with its end-to-end baseline. Yet, a larger label embedding vector size does not necessarily result in better performance. FC ($K = 1$) and FC ($K = 4$) has 3072 and 512-dimensional label embedding vectors respectively, but the bigger the embedding vector size, the worse their performances become.

The impacts of noise injection and $\mathcal{L}_{\text{feat}}$ are less obvious in Table 6, but the ablation experiments stress the importance of $\mathcal{L}_{\text{batch}}$. Dropping $\mathcal{L}_{\text{batch}}$ results in significant drops in accuracy, across all architecture. The poor performances without $\mathcal{L}_{\text{batch}}$ is in stark contrast to Wang et al. [2021], where $\mathcal{L}_{\text{feat}}$ alone maximizes the mutual information between intermediate features and labels $I(h, y)$. Without auxiliary networks to process intermediate features further, $I(h, y)$ obtained from $\mathcal{L}_{\text{feat}}$ alone seems limited. The experiments demonstrate that $\mathcal{L}_{\text{batch}}$, maximizing similarity between features and labels, extracts $I(h, y)$ much more effectively, even without auxiliary networks.

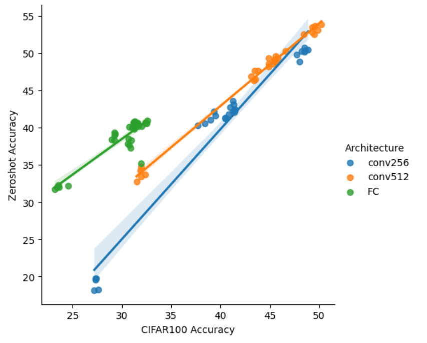


Figure 1: Correlation between zeroshot accuracy and CIFAR-100 fine-label prediction accuracy

Moreover, the use of dot product is crucial for $\mathcal{L}_{\text{batch}}$, as underlined by the severely impaired performance of the models with the cosine similarity on Table 2 and 6. Figure 2 illustrates how the dot product between feature vectors and label embedding vectors helps detect salient regions in images, even in early layers. Since we use the mean feature vector $\bar{\boldsymbol{h}}_n^l$ to represent each image in $\mathcal{L}_{\text{batch}}$ and Equation (Equation 1), prediction and feature extraction are based on the average dot product between feature vectors and label embedding vectors. Yet, the magnitude of salient vectors should contribute more to the dot product than that of non-salient feature vectors. If we normalize the magnitude as in the cosine similarity, we discard this valuable information. Thus, with the dot product, a small number of salient feature vectors can have more "votes" in prediction, as small regions of high attention are observed in some top-1 predictions in Figure 4n, 4f, 5b, 4b, and 3b.

Overall, $\mathcal{L}_{\text{batch}}$ and $\mathcal{L}_{\text{feat}}$ are best used in conjunction, as they generate the best performances together. Furthermore, features generated using both in Figure 7c are more disentangled than features generated using only one, as displayed in Figure 9c and 8c.

CIFAR-100 Coarse Labels		
Networks	Test error	std(%)
CFL conv256 (end-to-end)	42.38	
CFL conv512 (end-to-end)	40.46	
CFL FC end-to-end	59.02	
CFL Zeroshot conv256 ($\mathcal{L}_{\text{total-B}}$)	49.85	0.279
CFL Zeroshot conv256 ($\mathcal{L}_{\text{total-D}}$)	51.58	0.243
CFL Zeroshot conv256 (fixed $\boldsymbol{D}^Z$)	57.25	0.541
CFL Zeroshot conv256 (without ϵ)	58.23	0.465
CFL Zeroshot conv256 ($\mathcal{L}_{\text{batch}}$)	58.88	0.709
CFL Zeroshot conv256 ($\mathcal{L}_{\text{dict}}$)	WIP	WIP
CFL Zeroshot conv256 (cosine similarity)	80.92	0.761
CFL Zeroshot conv512 ($\mathcal{L}_{\text{total-B}}$)	**46.73**	0.377
CFL Zeroshot conv512 ($\mathcal{L}_{\text{total-D}}$)	47.04	0.54
CFL Zeroshot conv512 (fixed $\boldsymbol{D}^Z$)	50.64	0.497
CFL Zeroshot conv512 (without ϵ)	51.22	0.449
CFL Zeroshot conv512 ($\mathcal{L}_{\text{batch}}$)	53.03	0.534
CFL Zeroshot conv512 ($\mathcal{L}_{\text{dict}}$)	WIP	WIP
CFL Zeroshot conv512 (cosine similarity)	66.25	0.686
CFL Zeroshot FC ($\mathcal{L}_{\text{total-B}}$)	59.85	0.212
CFL Zeroshot FC ($\mathcal{L}_{\text{total-D}}$)	60.5	0.217
CFL Zeroshot FC (fixed $\boldsymbol{D}^Z$)	59.562	0.299
CFL Zeroshot FC ($K = 1$)	67.95	0.209
CFL Zeroshot FC ($K = 4$)	61.178	0.387
CFL Zeroshot FC (without ϵ)	59.76	0.418
CFL Zeroshot FC ($\mathcal{L}_{\text{batch}}$)	62.1	0.444
CFL Zeroshot FC ($\mathcal{L}_{\text{dict}}$)	WIP	WIP

Table 2: Zeroshot inference results on CIFAR-100 coarse-labels. The table lists the mean and standard deviation computed across five runs. Note that $\mathcal{L}_{\text{feat}}$ does not use $\boldsymbol{D}^Z$, so zeroshot inference is not possible with models trained only with $\mathcal{L}_{\text{feat}}$.

5 Discussion

Labels with similar label embedding vectors (in Figure 10a) tend to be close to each other in feature space. Label embedding vectors of animals are in general similar to each other. Likewise, animals are clustered to the left in Figure 7c. However, similarity between embedding vectors does not necessarily imply similarity between features or between prediction. For example, in Figure 10a, non-animal label embedding vectors are not similar to one another, but they are clustered to the right in Figure 7c. Moreover, dog and airplane embedding vectors are most similar, but they are not next to each other in feature space. Further, the confusion matrix in Figure 10b indicates that the models rarely misidentify labels of similar embedding vectors, except for dog and cat. This observation highlights that learning and inference rely on the interaction between features and label embedding vectors, rather than embedding vectors or features alone.

Biological Plausibility By eliminating BP, our method makes each local layer free of biologically implausible weight symmetry. Thus, each local layer can represent a non-symmetric, locally connected neuron compartment Guerguiev et al. [2016], Endo et al. [2021]. The label embeddings

	CIFAR-10		CIFAR-100	
CFL Architecture	Test error	std(%)	Test error	std(%)
FC ($\mathcal{L}_{\text{total-B}}$)	34.27	0.217	68.70	0.181
FC ($\mathcal{L}_{\text{total-D}}$)	33.78	0.238	67.71	0.261
FC (fixed $\boldsymbol{D}^Z$)	34.11	0.212	68.59	0.238
FC ($K = 1$)	43.34	0.185	76.32	0.449
FC ($K = 4$)	37.8	0.216	70.79	0.128
FC ($K = 12$)	35.21	0.097		
FC (without ϵ)	34.26	0.411	68.86	0.213
FC ($\mathcal{L}_{\text{batch}}$)	36.75	0.343	69.238	0.126
FC ($\mathcal{L}_{\text{dict}}$)	WIP	WIP	WIP	WIP
FC ($\mathcal{L}_{\text{feat}}$)	43.34	0.058	70.818	0.149
conv256 ($\mathcal{L}_{\text{total-B}}$)	24.39	0.137	51.71	0.476
conv256 ($\mathcal{L}_{\text{total-D}}$)	24.96	0.237	51.59	0.244
conv256 (fixed $\boldsymbol{D}^Z$)	25.10	0.131	58.71	0.166
conv256 (without ϵ)	25.34	0.142	59.08	0.709
conv256 ($\mathcal{L}_{\text{batch}}$)	26.31	0.504	61.21	0.646
conv256 ($\mathcal{L}_{\text{dict}}$)	WIP	WIP	WIP	WIP
conv256 ($\mathcal{L}_{\text{feat}}$)	44.16	0.488	73.24	0.163
conv256 (cosine simiarity)	40.64	0.712	72.62	0.138
conv512 ($\mathcal{L}_{\text{total-B}}$)	16.57	0.182	50.61	0.132
conv512 ($\mathcal{L}_{\text{total-D}}$)	16.66	0.128	50.5	0.675
conv512 (fixed $\boldsymbol{D}^Z$)	16.89	0.26	54.25	0.42
conv512 (without ϵ)	17.16	0.15	54.85	0.363
conv512 ($\mathcal{L}_{\text{batch}}$)	17.22	0.24	56.53	0.22
conv512 ($\mathcal{L}_{\text{dict}}$)	WIP	WIP	WIP	WIP
conv512 ($\mathcal{L}_{\text{feat}}$)	38.97	0.292	68.48	0.274
conv512 (cosine similarity)	36.14	0.556	68.08	0.273

Table 3: Ablation Results on CIFAR-10 and CIFAR-100. The table lists the mean and standard deviation computed across five runs. When training with cosine similarity, softmax sharpening temperature $\tau = 0.07$ was used.

can be considered as the high-order neural representation of the labels. Accordingly, $\mathcal{L}_{\text{batch}}$ mirrors the top-down interaction between the high-order label representation and the features in bottom-up processing layers Gilbert and Li [2013].

Our method also bears a resemblance to the models of pattern recognition in cognitive psychology: the template matching model, prototype matching, and feature matching models Shu-gen [2002]. According to each model, we recognize objects by comparing our perception of objects with templates, prototypes, or features, respectively. Label embedding vectors can be considered all templates, prototypes, and features; each label embedding vector embodies the prototypical feature of the label. The models without the linear classifier f^L on Table 1 predict labels by choosing the label with the highest dot product (1). Likewise, as shown in the dot product over the feature map (Figure 2), we may recognize the label whose embedding representation matches the image the most.

Conclusion In this work, we propose contrastive forward learning with label embeddings (CFL) as an alternative to the end-to-end training with BP. Unlike local learning which still leverages BP in auxiliary networks, CFL completely eliminates BP and therefore the biologically implausible weight symmetry. Freedom from BP makes CFL models completely forward and backward unlocked, allowing each layer to update its weights after a forward pass. Unlike existing local and forward learning approaches, CFL can leverage BP architectures with minimal modification, requiring no specialized architecture, auxiliary network, input distortion, or local projection. Furthermore, the biologically plausible label embedding grants interpretability and allows for zeroshot inference through interpolation of label-specific representation. In addition to the above, CFL outperforms existing forward learning approaches on non-trivial datasets such as CIFAR-10 and CIFAR-100.

References

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

Paul J. Werbos. Beyond regression : "new tools for prediction and analysis in the behavioral sciences. 1974.

Timothy P. Lillicrap, Daniel Cownden, Douglas Blair Tweed, and Colin J. Akerman. Random feedback weights support learning in deep neural networks. *ArXiv*, abs/1411.0247, 2014.

Timothy P. Lillicrap, Daniel Cownden, Douglas Blair Tweed, and Colin J. Akerman. Random synaptic feedback weights support error backpropagation for deep learning. *Nature Communications*, 7, 2016.

Arild Nøkland. Direct feedback alignment provides learning in deep neural networks. In *NIPS*, 2016.

Charles Gilbert and Wu Li. Top-down influences on visual processing. *Nature Reviews Neuroscience*, 14:350–363, 2013.

Dong-Hyun Lee, Saizheng Zhang, Asja Fischer, and Yoshua Bengio. Difference target propagation. In *ECML/PKDD*, 2014.

Arild Nøkland and Lars Hiller Eidnes. Training neural networks with local error signals. *ArXiv*, abs/1901.06656, 2019.

Eugene Belilovsky, Michael Eickenberg, and Edouard Oyallon. Decoupled greedy learning of cnns. *ArXiv*, abs/1901.08164, 2019.

Charlotte Frenkel, Martin Lefebvre, and David Bol. Learning without feedback: Fixed random learning signals allow for feedforward training of deep neural networks. *Frontiers in Neuroscience*, 15, 2021.

Geoffrey Hinton. The forward-forward algorithm: Some preliminary investigations, 2022.

Charlotte Frenkel, J. D. Legat, and David Bol. A 28-nm convolutional neuromorphic processor enabling online learning with spike-based retinas. *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5, 2020.

Tomas Mikolov, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *International Conference on Learning Representations*, 2013.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018. URL `http://arxiv.org/abs/1810.04805`.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. A simple framework for contrastive learning of visual representations. *ArXiv*, abs/2002.05709, 2020a.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. A simple framework for contrastive learning of visual representations. *ArXiv*, abs/2002.05709, 2020b.

Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. *ArXiv*, abs/2004.11362, 2020.

Aäron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *ArXiv*, abs/1807.03748, 2018.

Taesung Park, Alexei A. Efros, Richard Zhang, and Jun-Yan Zhu. Contrastive learning for unpaired image-to-image translation. In *European Conference on Computer Vision*, 2020.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, 2021.

Pierre Baldi and Peter Sadowski. A theory of local learning, the learning channel, and the optimality of backpropagation. *Neural networks : the official journal of the International Neural Network Society*, 83:51–74, 2015.

Eugene Belilovsky, Michael Eickenberg, and Edouard Oyallon. Greedy layerwise learning can scale to imagenet. *ArXiv*, abs/1812.11446, 2018.

P. Pathak, Jingwei Zhang, and Dimitris Samaras. Local learning on transformers via feature reconstruction. *ArXiv*, abs/2212.14215, 2022.

Yulin Wang, Zanlin Ni, Shiji Song, Le Yang, and Gao Huang. Revisiting locally supervised learning: an alternative to end-to-end training. *ArXiv*, abs/2101.10832, 2021.

Michael U Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *International Conference on Artificial Intelligence and Statistics*, 2010.

Alexander Ororbia and Ankur Arjun Mali. The predictive forward-forward algorithm. *ArXiv*, abs/2301.01452, 2023.

Rajesh P. N. Rao and Dana H. Ballard. Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nature Neuroscience*, 2:79–87, 1999a.

Tommaso Salvatori, Yuhang Song, Yujian Hong, Simon Frieder, Lei Sha, Zhenghua Xu, Rafał Bogacz, and Thomas Lukasiewicz. Associative memories via predictive coding. *Advances in neural information processing systems*, 34:3874–3886, 2021.

Alexander Ororbia and Daniel Kifer. The neural coding framework for learning generative models. *Nature Communications*, 13, 2020.

Rajesh P. N. Rao and Dana H. Ballard. Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nature Neuroscience*, 2:79–87, 1999b.

Heung-Chang Lee and Jeong geun Song. Symba: Symmetric backpropagation-free contrastive learning with forward-forward algorithm for optimizing convergence. *ArXiv*, abs/2303.08418, 2023.

James Kirkpatrick, Razvan Pascanu, Neil C. Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114:3521 – 3526, 2016.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017. URL `http://arxiv.org/abs/1706.03762`.

Praveen Badimala, Chinmaya Mishra, Reddy Kumar Modam Venkataramana, Syed Bukhari, and Andreas Dengel. A study of various text augmentation techniques for relation classification in free text. pages 360–367, 02 2019. doi: 10.5220/0007311003600367.

Qiying Yu, Jieming Lou, Xianyuan Zhan, Qizhang Li, Wangmeng Zuo, Yang Liu, and Jingjing Liu. Adversarial contrastive learning via asymmetric infonce. *ArXiv*, abs/2207.08374, 2022.

Yann LeCun and Corinna Cortes. The mnist database of handwritten digits. 2005.

Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009.

Gongpei Zhao, Tao Wang, Yidong Li, Yi Jin, Congyan Lang, and Haibin Ling. The cascaded forward algorithm for neural network training. *ArXiv*, abs/2303.09728, 2023.

Jordan Guerguiev, Timothy P. Lillicrap, and Blake A. Richards. Towards deep learning with segregated dendrites. *eLife*, 6, 2016.

Masaaki Endo, Hisato Maruoka, and Shigeo Okabe. Advanced technologies for local neural circuits in the cerebral cortex. *Frontiers in Neuroanatomy*, 15, 2021.

Wang Shu-gen. Framework of pattern recognition model based on the cognitive psychology. *Geo-spatial Information Science*, 5:74–78, 2002.

Appendix A Implementation Details and Visualization

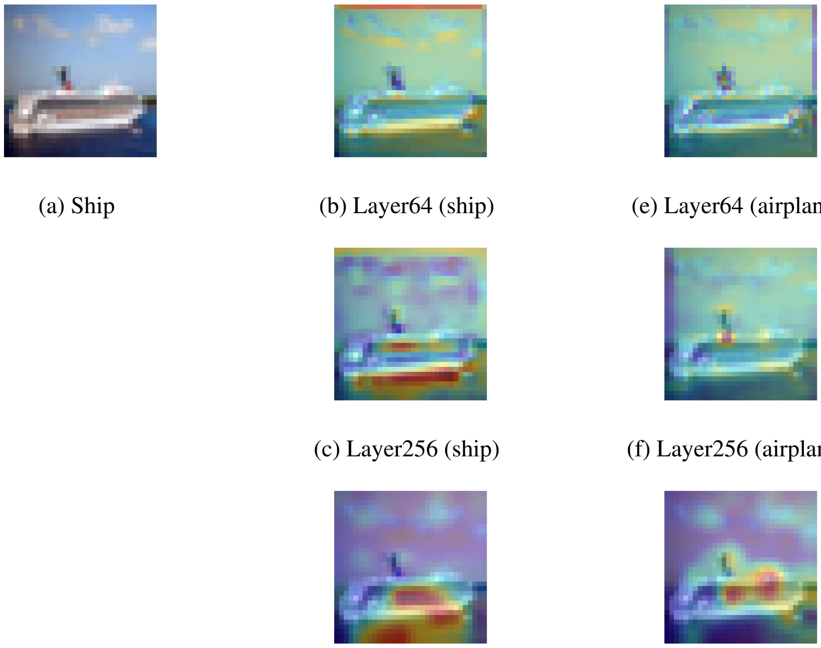
	FC (MNIST)	FC (CIFAR-10)	FC (CIFAR-100)	conv256	conv512
$\boldsymbol{D}^Z$	10x98	10x128	100x256	10X256 (100x256)	10x512 (100x512)
K	8	24	12		
Input Size	784	3072	3072	3x32x32	3x32x32
1	FC 1024	FC 3072	FC 3072	CONV(3x3x64,1,0)	CONV(3x3x64,1,0)
2	ReLU	ReLU	ReLU	Batchnorm	Batchnorm
3	Layernorm	Layernorm	Dropout 0.3	ReLU	ReLU
4	FC 1024	FC 3072	LayerNorm	CONV(3x3x256,2,1)	CONV(3x3x256,2,1)
5	ReLU	ReLU	FC3072	Batchnorm	Batchnorm
6	Layernorm	Layernorm	ReLU	ReLU	ReLU
7	FC 10	FC 10	Dropout 0.3	Global Avg Pooling	CONV(3x3x512,2,1)
8			Layernorm	FC 10 (100)	Batchnorm
9			FC 100		ReLU
10					Global Avg Pooling
11					FC 10 (100)

Table 4: Architectures used in the experiments. CFL and BP share the same architecture, except for $\boldsymbol{D}^Z$. CFL (without f^L) and BP ($\mathcal{L}_{\text{total}}^{L-1}$) both share the same architecture, devoid of the final FC layer.

	Epochs	Learning Rate	Decay Milestones
MNIST FC	150	0.0005	50 100 125
CIFAR-10 FC	400	0.0002	50 150 200 350
CIFAR-100 FC	200	0.0001	50 100 150
MNIST conv	150	0.0075	50 75 100 125
CIFAR-10 conv	500	0.0075	100 200 300 400 450
CIFAR-100 conv256	400	0.0075	100 200 250 300 350
CIFAR-100 conv512	200	0.0075	50 75 100 125 150

	Optimizer	Batch size	Learning rate decay rate
All experiments	AdamW	512	0.5

Table 5: Hyperparameters.



(a) Ship (b) Layer64 (ship) (e) Layer64 (airplane)

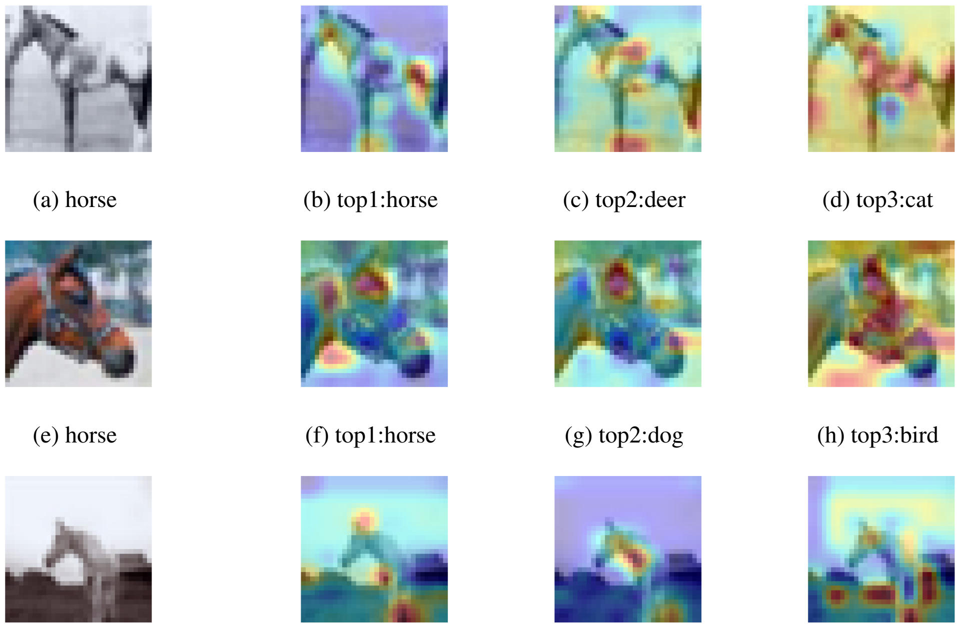
(c) Layer256 (ship) (f) Layer256 (airplane)

(d) Layer512 (ship) (g) Layer512 (airplane)

Figure 2: Attention maps of the dot product between intermediate layer features and label embedding vectors. The redness denotes high values, whereas the blueness represents low values. 2a is the ground truth image and label. The column 2b, 2c, and 2d corresponds to the top-1 predicted label, whereas the column 2e, 2f, and 2g corresponds to the top-2 predicted label. With the "ship" embedding vector, more attention is given to the water and the hull of the ship. The attention to the hull starts at Layer256. In contrast, with the "airplane" embedding vector, more attention is given to the upper body of the ship and the sky.

	CIFAR-10	
CFL Architecture	Params	Test error
<u>ViT BP</u>	3.19M	23.27
CFL ViT ($\mathcal{L}_{\text{total-B}}$)	3.22M	37.73
DRTP FC (Frenkel et al. [2021])	4.09M	51.06
DRTP CONV (Frenkel et al. [2021])	16.54M	30.59
FF (Hinton [2022])	18.93M	41
CAFO (Zhao et al. [2023]	243K	32.6
PFF (Ororbia and Mali [2023])	32.37M	50.14*
Symba (Lee and geun Song [2023])	16M	41.2
Symba (Lee and geun Song [2023])	31M	40.9

Table 6: Preliminary Results of ViT on CIFAR-10

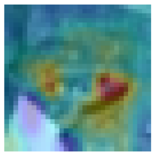
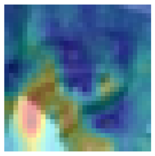
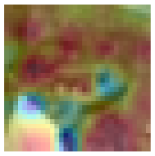


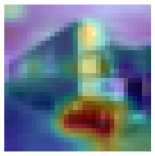
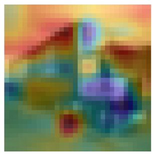
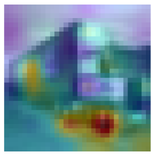
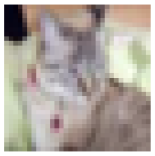
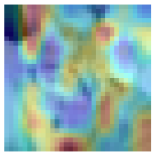
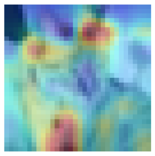
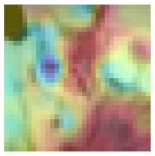
(a) horse (b) top1:horse (c) top2:deer (d) top3:cat

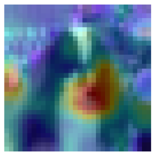
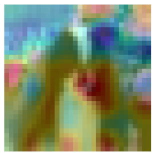
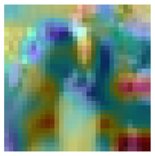
(e) horse (f) top1:horse (g) top2:dog (h) top3:bird

(i) horse (j) top1:deer (k) top2:horse (l) top3:cat

Figure 3: Attention maps for horse images. The third row represents the misclassification case.

(a) frog

(b) top1:frog

(c) top2:cat

(d) top3:bird

(e) truck

(f) top1:truck

(g) top2:airplane

(h) top3:automobile

(i) cat

(j) top1:cat

(k) top2:ship

(l) top3:automobile

(m) dog

(n) top1:dog

(o) top2:cat

(p) top3:bird

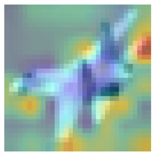
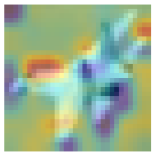
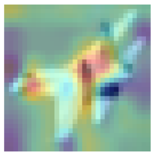
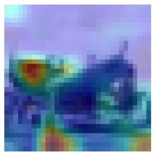
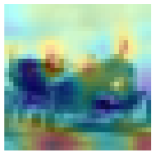
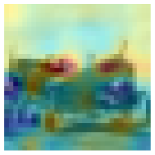
(q) airplane

(r) top1:airplane

(s) top2:bird

(t) top3:horse

Figure 4: Attention maps for the correct top-1 classification cases.

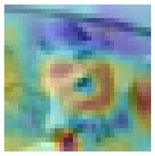
(a) airplane

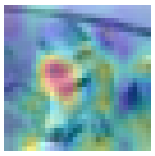

(b) top1:truck

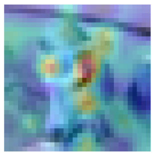

(c) top2:airplane

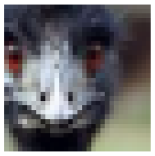

(d) top3:ship

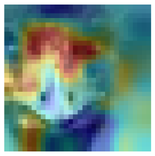
(e) deer

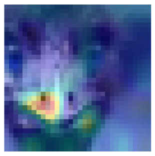

(f) top1:dog

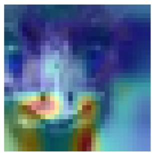

(g) top2:cat


(h) top3:deer

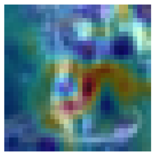

(i) bird

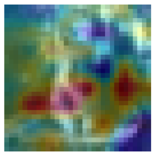

(j) top1:cat

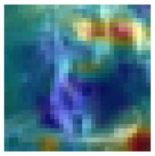

(k) top2:ship


(l) top3:automobile

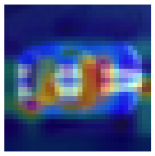
(m) cat

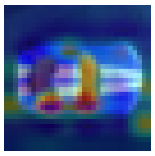

(n) top1:dog


(o) top2:cat


(p) top3:automobile

(q) automobile


(r) top1:truck


(s) top2:automobile

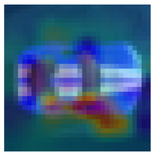
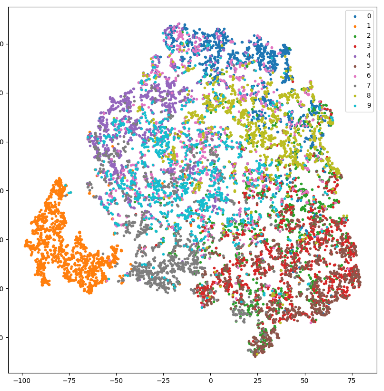
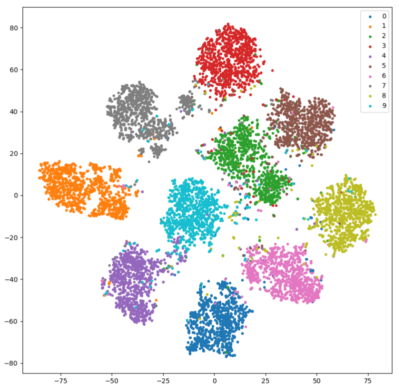

(t) top3:ship

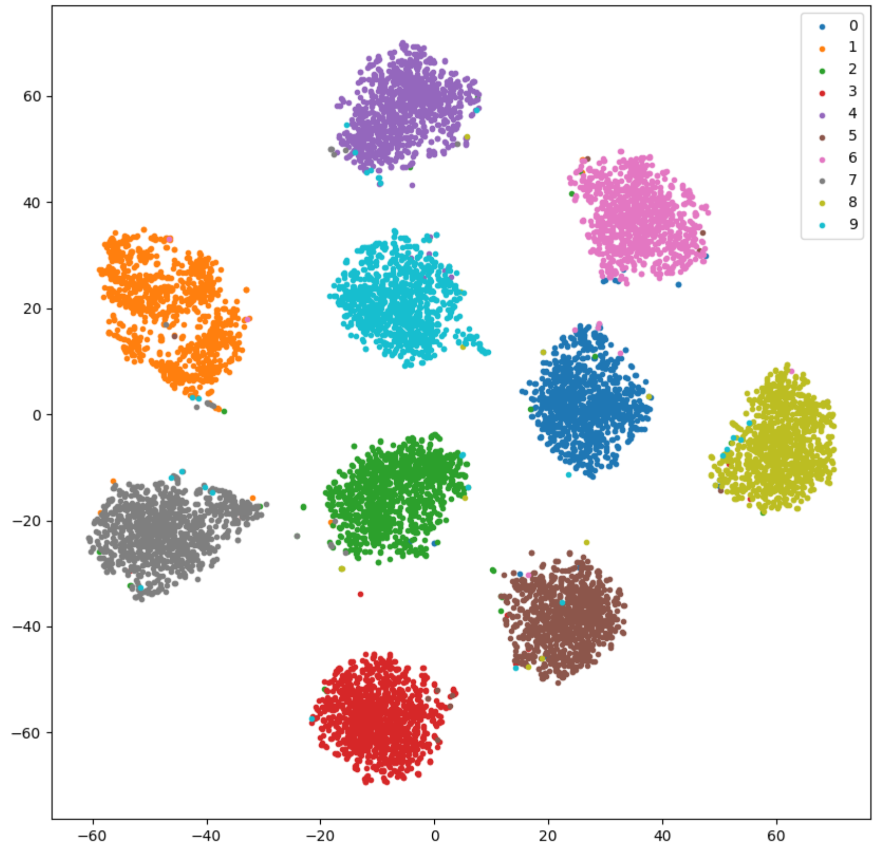
Figure 5: Attention maps for the top-1 misclassification cases.



(a) 64 dimensional features from the first layer

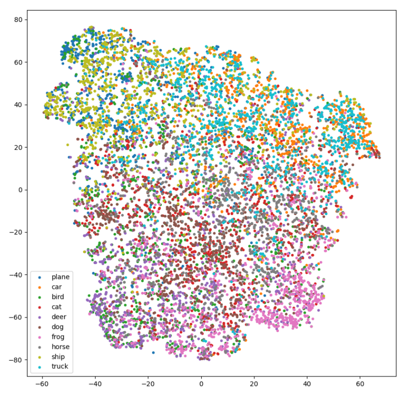


(b) 256 dimensional features from the second layer

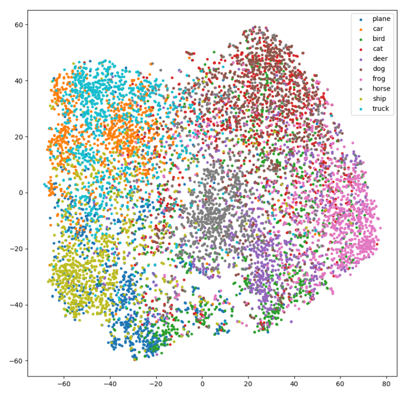


(c) 512 dimensional features from the third layer

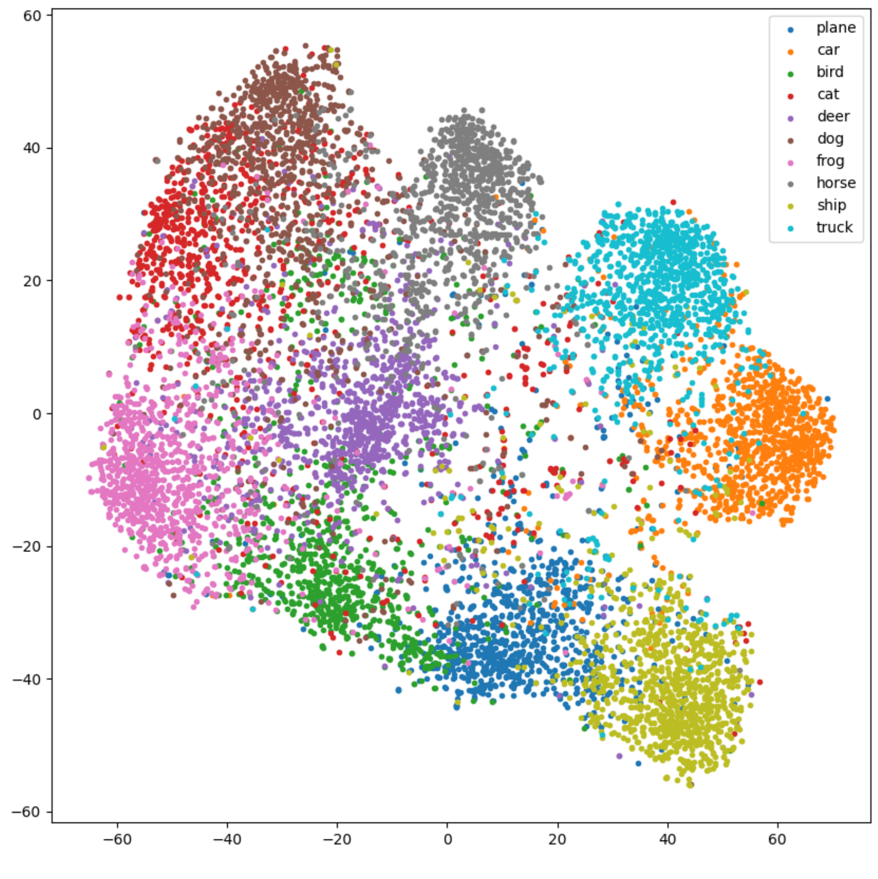
Figure 6: t-SNE of MNIST features from the CFL conv512 model.



(a) 64 dimensional features from the first layer

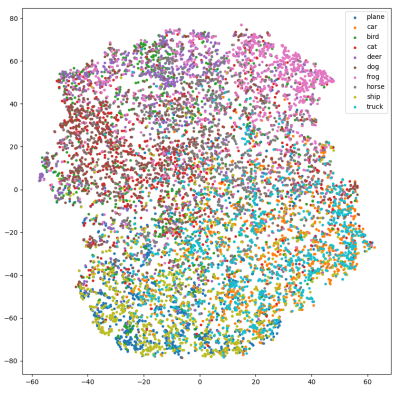


(b) 256 dimensional features from the second layer

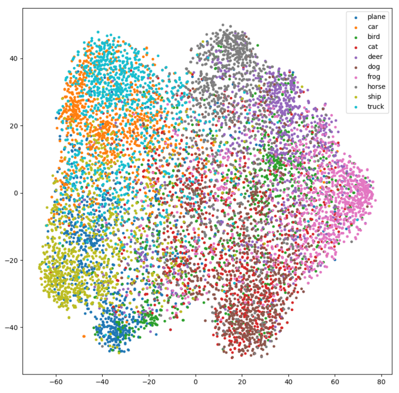


(c) 512 dimensional features from the third layer

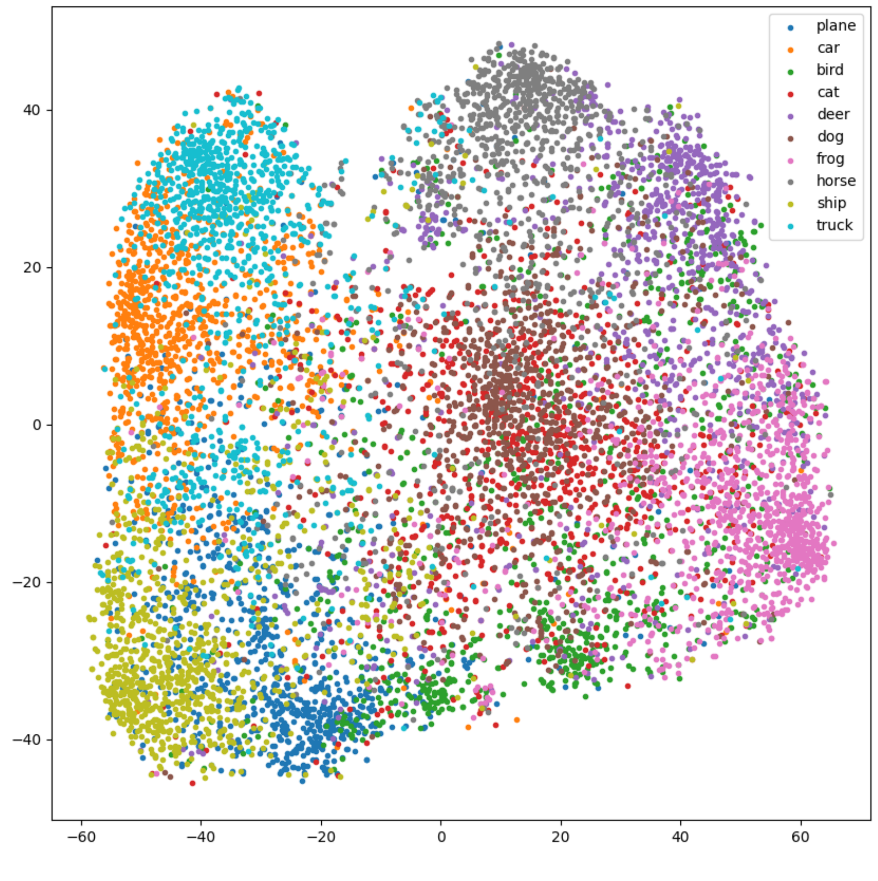
Figure 7: t-SNE of CIFAR10 features from the CFL conv512 model. Animals are clustered to the left, while non-animals are clustered to the right.



(a) 64 dimensional features from the first layer

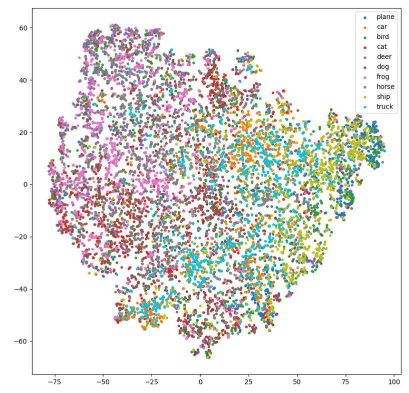


(b) 256 dimensional features from the second layer

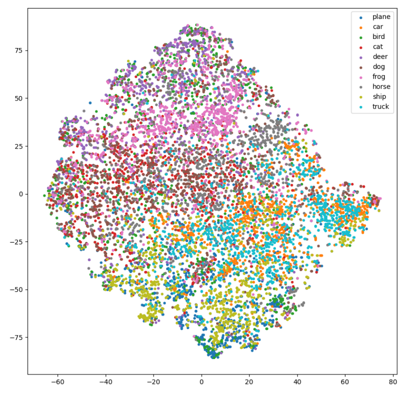


(c) 512 dimensional features from the third layer

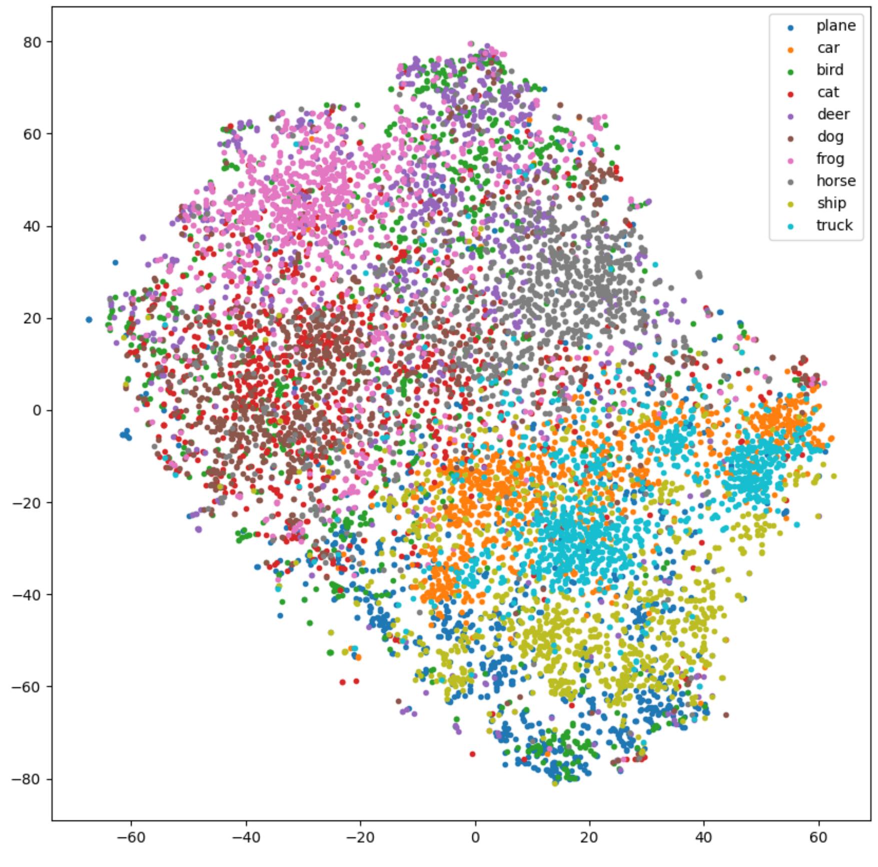
Figure 8: t-SNE of CIFAR10 features from the CFL conv512 ($\mathcal{L}_{\text{feat}}$) model.



(a) 64 dimensional features from the first layer

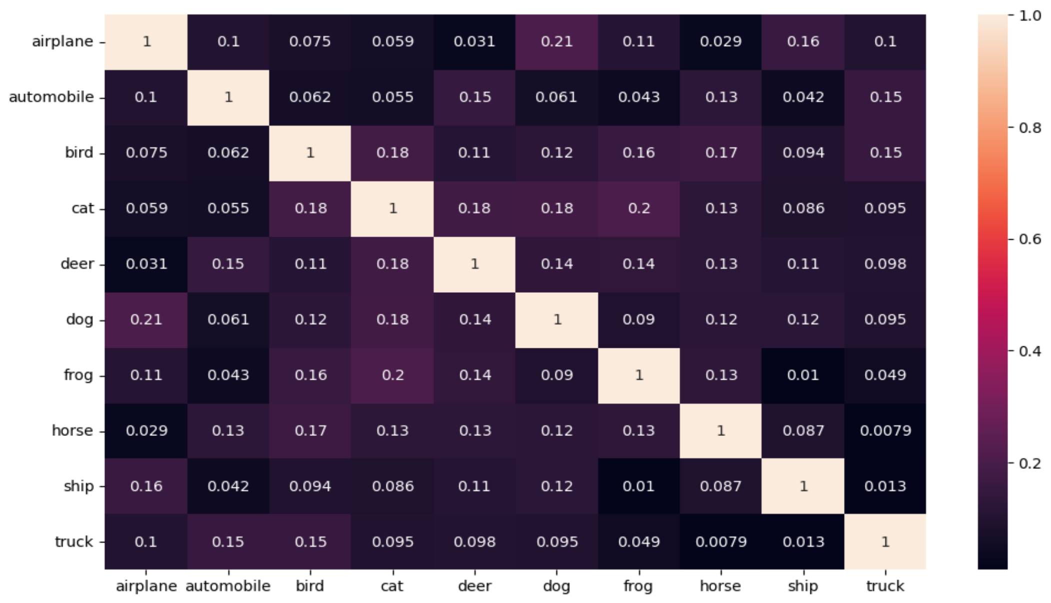


(b) 256 dimensional features from the second layer

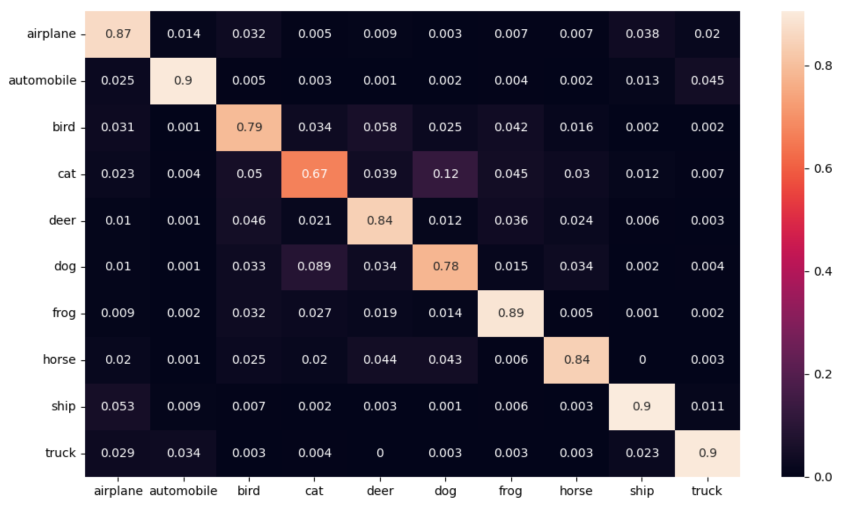


(c) 512 dimensional features from the third layer

Figure 9: t-SNE of CIFAR10 features from the CFL conv512 ($\mathcal{L}_{\text{batch}}$) model. Without $\mathcal{L}_{\text{feat}}$, features are less disentangled than those in Figure 7c.



(a) Label Embedding Similarity Matrix



(b) Prediction Confusion Matrix

Figure 10: Similarity and Confusion Matrix for CFL conv512 on CIFAR-10