Deep Companion Learning: Enhancing Generalization Through Historical Consistency

Ruizhao Zhu and Venkatesh Saligrama

Boston University, Boston MA, 02215, USA
{rzhu,srv}@bu.edu

Abstract. We propose Deep Companion Learning (DCL), a novel training method for Deep Neural Networks (DNNs) that enhances generalization by penalizing inconsistent model predictions compared to its historical performance. To achieve this, we train a deep-companion model (DCM), by using previous versions of the model to provide forecasts on new inputs. This companion model deciphers a meaningful latent semantic structure within the data, thereby providing targeted supervision that encourages the primary model to address the scenarios it finds most challenging. We validate our approach through both theoretical analysis and extensive experimentation, including ablation studies, on a variety of benchmark datasets (CIFAR-100, Tiny-ImageNet, ImageNet-1K) using diverse architectural models (ShuffleNetV2, ResNet, Vision Transformer, etc.), demonstrating state-of-the-art performance.

1 Introduction

Stochastic Gradient Descent (SGD) underpins Deep Neural Networks (DNNs) training at scale. We propose a novel training method to improve SGD generalization in the context of supervised learning. While there are a number of prior works that have focused on improving SGD generalization, as we point out later, our perspective is shaped by the view that SGD is inherently stochastic[1], and we posit that controlling the variability of SGD trajectory during training can improve generalization. There are two sources of variability:

(a) Due to variability across different batches, a model can observe a wide deviation in empirical losses on two different batches, and as such the observed loss on any batch may not be indicative of the true loss.
(b) Due to the randomness of batches, the SGD model trajectory is stochastic, and so on a new batch, the model prediction can be widely diverging depending on which SGD model was realized prior to that round.

Our focus in this paper is on (b) as we believe the solution to (a) can benefit from more data. While data augmentation can serve as a solution for improving generalization, we view that effort as complementary to our study. To account

[1] Due to random initialization and random choices involved in batch processing [9, 33].

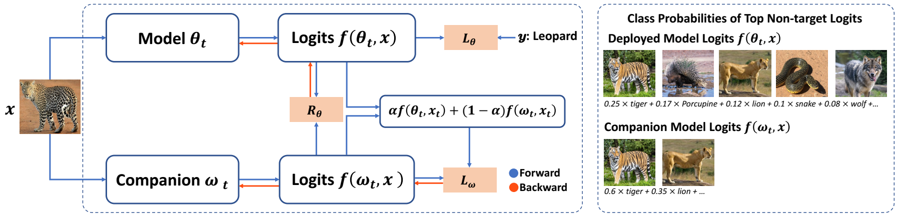
x
Model θ_t
Logits $f(\theta_t, x)$
L_θ
y: Leopard
R_θ
$\alpha f(\theta_t, x_t) + (1-\alpha) f(\omega_t, x_t)$
Companion ω_t
Logits $f(\omega_t, x)$
L_ω
● Forward
● Backward
Class Probabilities of Top Non-target Logits
Deployed Model Logits $f(\theta_t, x)$
0.25 × tiger + 0.17 × Porcupine + 0.12 × lion + 0.1 × snake + 0.08 × wolf +...
Companion Model Logits $f(\omega_t, x)$
0.6 × tiger + 0.35 × lion + ...

Fig. 1: Method Overview. At iteration t, we optimize the instantaneous model $\boldsymbol{\theta}_t$ (model eventually deployed upon training) with standard cross entropy loss and a regularizer enforcing consistency with model $\boldsymbol{\omega}$. Model $\boldsymbol{\omega}$ is recursively updated by approximating predictions from its previous embodiment and the predictions of current model $\boldsymbol{\theta}_t$. (Right) Probability of class as the top non-target class is shown. The companion model helps narrow down the top non-target classes as tiger and lion for class leopard. In the initial training stage, the top non-target of deployed model logits are more randomly distributed with some irrelevant classes. The companion model can help capture a general semantic structure of the dataset.

for (b) we propose to penalize predictions made by model updates that deviate significantly from our forecast. To achieve this, we train a deep-companion model (DCM), by using previous versions of the model to provide forecasts on new inputs. In this context, our goal poses two fundamental challenges:

(i) How to best utilize past history to forecast outputs on new input examples?
(ii) How can we learn to make predictions efficiently?

The challenges in (i) includes: (a) designing a good look-back horizon to balance recency with historical trends, and how to use these trends to forecast; (b) what is the latent space where such predictions make most sense. The challenge in (ii) requires that our method does not significantly expand the storage or computational footprint of vanilla SGD.

Deep Companion Model. We address (ii) through a companion neural network that aims to identify a prediction that minimizes the disagreement between itself and the preceding models. This companion network mirrors the architecture of the primary model (ablations with smaller networks appear in supplementary) currently undergoing training. We perform analogous SGD steps to train the companion model aligning its optimization process with that of the primary model. We use an exponential smoothing parameter and hyperparameter tune it to optimize the look-back horizon. We supervise the companion in the logit space (before softmax) by minimizing the mean-squared error. Intuitively, this makes sense because we expect well-clustered and linearly separable features in the logit space.

Enforcing Predictive Consistency. Our proposal is depicted in Fig. 1. We measure the difference between DCM output, $f(\boldsymbol{\omega}_t, \mathbf{x}_t)$ and SGD model, $f(\boldsymbol{\theta}_t, \mathbf{x}_t)$, and use this difference as a penalty. Our intuition here is related to the notion of cumulative regret, a concept arising in streaming settings [36]. Although our measure is not a true measure of regret[2], the DCM output can be viewed as a prediction on a new batch without the benefit of hindsight, while the SGD model output reflects the best achievable with hindsight of the new batch data. Intuitively, significant deviations are likely a result of SGD overfitting to the current batch, and performance can be improved by penalizing inconsistency. A different perspective is depicted in Figure 1 (right). The companion model consistently outputs the same top non-target class, capturing a generalizable semantic structure of the dataset. In particular, for the class Leopard, the companion model has class Tiger or Lion as the top non-targets.

Experimental Results. We run experiments on several benchmark datasets (CIFAR-100, TinyImageNet, ImageNet-1K) and architectures (ShuffleNetV2, Resnet-18, Resnet-50, and ViT-Tiny) and show that our proposed input and model consistency proxies lead to improved SOTA performance. In particular, on CIFAR-100, our results, obtained without any pre-training, attain performance gains larger than those that utilize pre-training. In general, pre-training results in a computational bottleneck while adapting to target data. Our proposed method suggests that these bottlenecks can be overcome through a better-chosen training scheme. Our method also scales to ImageNet-1k dataset with transformer based architecture. Addtionally, DCL demonstrates its potential as a plug-and-play technique in various applications such as fine-tuning, self-supervised pre-training, and semi-supervised learning.

Contributions. The main contributions of our paper are:

- *Efficient Consistency Predictor.* We propose a computationally efficient method that uses predictions of previous model versions to forecast consistent outputs on new examples. The companion model infers a meaningful semantic structure.
- *Data-Dependent Dynamic Regularization.* Our regularizer penalizes deviations of its predictions from the companion model predictions. Since the companion model is updated in parallel, the regularization induced is dynamic, and since it penalizes predictions rather than parameters, it is data-dependent.
- *Improved Representation.* Our choice of logit space enforces better linearly separability of different classes resulting in better representations based on effectively inferring the underlying semantic structure.
- *Empirical Results.* We demonstrate SOTA performance on diverse benchmark datasets and architectures. We show that training from scratch achieves similar accuracies as models with pre-training, thereby overcoming the computational overhead of adapting pre-training to target data.

2 Related Work

Various regularization techniques have been employed to enhance model generalization. These include data augmentations [8, 10, 18, 32, 43, 44], dropout regularization [38], normalization [1, 21, 42] and penalty functions [26, 30].

[2] every input sample in our case has been previously observed

Parameter Regularization Penalty functions, in particular, are integrated with the primary loss function and jointly optimized. These functions are carefully crafted to induce specific properties to the loss function. For example, penalizing L^2-norm on parameters is traditionally adopted to mitigate overfitting. Furthermore, modifications to the loss landscape geometry have been proposed, with some regularizers targeting sharpness [12, 13, 45]. These regularizers capture specific local properties of the loss landscape over the parameter space, and as such are data-agnostic. We propose a method that enforces penalty on the predicted outputs, and thus inducing a data-dependent regularization.

Data-Dependent Regularization Our method is most closely related to prior works that explicitly or implicitly induce data-dependent regularization. Variance reduction methods [9] propose to reduce variance by using a control variate derived from a previously stored anchor model. While the idea of variance reduction is related to ours, they are evidently ineffective for deep models [9]. In contrast, we train a companion model that is continuously updated to provide consistent forecasts, and as such is more effective. Similar to our approach, [22] aims to reduce variance by averaging over stochastic gradient models. Other methods, such as [3, 4, 7, 13, 14, 17, 20, 40], focus on achieving consistency across different model views. Specifically, Mean Teacher [40] uses an exponential moving average (EMA) of model parameters as an anchor to interact with the current model. Unlike our method, which combines predictions, EMA fuses model parameters directly. Temporal Ensembling [27] uses EMA predictions for each training instance at each epoch. PS-KD [23] employs a self-distillation approach, utilizing a previous model as a teacher to provide soft labels for the student model. Unlike us, they do not update the teacher model. Self-supervised learning methods [3, 14] propose similar penalties in the absence of ground truth. These works manipulate the loss landscape in parameter space like EMA [22, 40], differing from our approach. DML [46] trains two models simultaneously with different initializations, each model alternately serving as a teacher with pseudo-labels. In contrast, we start with a single initialization and learn a companion model to capture the mean behavior in the logit space, reshaping the feature representation for the deployed model to be more compactly clustered. Our regularizer adapts to both historical and recent predictions, creating a surrogate for controlling variability.

3 Method

In this section, we propose Deep Companion Learning (DCL) that utilizes a regularizer to enforce prediction consistency during SGD training. First, we describe our method in Section 3.1 and the algorithm and pseudo-code in Section 3.3. Subsequently, in Section 3.4 we present an intuitive justification for our method.

3.1 Deep Companion Learning Method

Notation. For simplicity, we will focus on a $K-$class classification problem with $\mathcal{X}$ and $\mathcal{Y}$ being the input and output spaces respectively. A training set of N

i.i.d. data points $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^N$ sampled from a joint distribution P on $\mathcal{X} \times \mathcal{Y}$ is provided, where $\mathbf{x}_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$. Let P^N be a distribution of any N-sample dataset. We parameterize the neural network with parameters $\boldsymbol{\theta}$. Given an data sample $(\mathbf{x}_i, y_i)$, where $\mathbf{x}_i \in \mathcal{X}, \mathbf{y}_i \in \mathcal{Y}$, $f(\boldsymbol{\theta}, \mathbf{x}_i)$ denotes the network output logits, and $\ell\left(f(\boldsymbol{\theta}, \mathbf{x}_i), y_i\right)$ denotes the loss. We consider the empirical risk minimization as $\min_{\boldsymbol{\theta}} \frac{1}{N} \sum_i \ell\left(f(\boldsymbol{\theta}, \mathbf{x}_i), y_i\right) := \min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta})$. We construct another network with the same architecture, but parameterized differently with $\boldsymbol{\omega}$. Similarly, $f(\boldsymbol{\omega}, \mathbf{x}_i)$ represents the output logits for input $\mathbf{x}_i$ under the $\boldsymbol{\omega}$ model. We refer to $\boldsymbol{\omega}$ as a companion model. Subscript t represents the step of the iteration.

To build intuition into our method, let us consider a data sample as a triple, $(\mathbf{x}_i, y_i, \mathbf{z}_i)$ consisting of the input $\mathbf{x}_i$, the ground-truth label y_i, and an auxiliary observation $\mathbf{z}_i$, for instance, logits obtained from an auxiliary network on the input $\mathbf{x}_i$. What would be our goal in this case? Naturally, we would like to train our model $\boldsymbol{\theta}$ by including the auxiliary observation as supervision. To this end, we define a new loss, $\Delta(f(\boldsymbol{\theta}, \mathbf{x}_i), \mathbf{z}_i)$, which denotes the distance between logits predicted by $\boldsymbol{\theta}$ and the auxiliary observation. As such our global objective $R(\boldsymbol{\theta})$ is to optimize:

$$R(\boldsymbol{\theta}) = \frac{1}{N} \sum_{i=1}^{N} r(\boldsymbol{\theta}, \mathbf{x}_i, y_i, \mathbf{z}_i) \triangleq \frac{1}{N} \sum_{i=1}^{N} [\ell(f(\boldsymbol{\theta}, \mathbf{x}_i), y_i) + \lambda \Delta(f(\boldsymbol{\theta}, \mathbf{x}_i), \mathbf{z}_i)] \tag{1}$$

where λ is a hyperparameter.

Let us now describe SGD in this context. In round t nature chooses an example $(\mathbf{x}_t, y_t, \mathbf{z}_t)$ uniformly at random from the dataset, and a corresponding risk function $r_t(\boldsymbol{\theta}) = r(\boldsymbol{\theta}, \mathbf{x}_t, y_t, \mathbf{z}_t)$. SGD then takes a gradient step on the observed risk, namely,

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \nabla_{\boldsymbol{\theta}} (\ell(f(\boldsymbol{\theta}_t, \mathbf{x}_i), y_i) + \lambda \Delta(f(\boldsymbol{\theta}_t, \mathbf{x}_t), \mathbf{z}_t) \tag{2}$$

3.2 Companion Model

Let us now discuss training a companion model, $\boldsymbol{\omega}$ to predict auxiliary logits, $\mathbf{z} = f(\boldsymbol{\omega}, \mathbf{x})$, which serve as supervision for our deployed model update above.At round t, we have historical information upto round $t-1$, and the goal of an auxiliary model is to offer a forecast for the logits corresponding to the new input, $\mathbf{x}_t$ in round t. As such it makes sense for the companion model to provide supervision that complements the ground truth information y_t. A direct choice is to encourage the companion model to be close to all the historical models $\{\boldsymbol{\theta}_t, \boldsymbol{\theta}_{t-1}, ..., \boldsymbol{\theta}_1\}$, we design the companion objective as in Equation 3.

$$\boldsymbol{\omega}_t = \arg\min_{\boldsymbol{\omega}} \left(\sum_{i=1}^{t} \Delta(f(\boldsymbol{\omega}, \mathbf{x}_t), f(\boldsymbol{\theta}_i, \mathbf{x}_t)) \right) \tag{3}$$

The update rule in 3 requires all historical models, which means that the memory complexity grows linearly as training iterates, and thus impractical. When the distance function is MSE, i.e. $\mathit{\Delta}(f(\boldsymbol{\theta},\mathbf{x}), f(\boldsymbol{\omega}_t,\mathbf{x})) = \frac{1}{2}\|f(\boldsymbol{\theta},\mathbf{x}) - f(\boldsymbol{\omega}_t,\mathbf{x})\|^2$, the well-known orthogonality property leads to the following observation:

$$\frac{1}{t}\sum_{i=1}^{t}\|f(\boldsymbol{\theta}_i,\mathbf{x}) - f(\boldsymbol{\omega},\mathbf{x})\|^2 = \|f(\boldsymbol{\omega},\mathbf{x}) - \frac{1}{t}\sum_{i=1}^{t} f(\boldsymbol{\theta}_i,\mathbf{x})\|^2 + \mathrm{Var}(f(\boldsymbol{\theta}_i,\mathbf{x}))$$

The term $\frac{1}{t}\sum_{i=1}^{t} f(\boldsymbol{\theta}_i,\mathbf{x})$ can be expressed recursively assuming the companion model in the previous round is a good approximation:

$$\frac{1}{t}\sum_{i=1}^{t} f(\boldsymbol{\theta}_i,\mathbf{x}) = \frac{t-1}{t} f(\boldsymbol{\omega}_{t-1},\mathbf{x}) + \frac{1}{t} f(\boldsymbol{\theta}_t,\mathbf{x}) + \text{noise}$$

Ignoring the noise term, and substituting hyperparameters α in place of $\frac{t-1}{t}$, $\frac{1}{t}$ we can replace Equation 3 with the following objective:

$$\boldsymbol{\omega}_t = \arg\min_{\boldsymbol{\omega}} \frac{1}{2}\|f(\boldsymbol{\omega},\mathbf{x}), \alpha f(\boldsymbol{\omega}_{t-1},\mathbf{x}) + (1-\alpha) f(\boldsymbol{\theta}_t,\mathbf{x})\|^2 \tag{4}$$

The objective then reduces to aligning the logits of the current companion model with a convex combination of the output of the previous companion model, and the output of the instantaneous deployed model.

3.3 Implementation

Algorithm. The end-to-end pseudo code is displayed in Algorithm 1. Deep Companion Learning (DCL) algorithm, iteratively trains the (deployed) model as in Equation 2 and the companion model as in Equation 4, by taking a gradient step on the batch data.

Algorithm 1 Deep Companion Learning (DCL)

```
Input: Training data $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$
Parameters: Iteration $T$, batch size $B$, learning rates $\eta_\theta$, $\eta_\omega$, companion weight $\alpha$
Initialize: Randomly initialize model $\boldsymbol{\theta}_0$, $\boldsymbol{\omega}_0$ with the same initialized parameters.
for $t = 0$ to $T - 1$ do
    Sample a batch of data $\{\mathbf{x}, \mathbf{y}\}$
    Update the instantaneous deployed model:
    $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta_{\boldsymbol{\theta}} \nabla_{\boldsymbol{\theta}}(\mathcal{L}(\boldsymbol{\theta}) + \mathit{\Delta}(f(\boldsymbol{\theta},\mathbf{x}), f(\boldsymbol{\omega}_t,\mathbf{x})))$
    Update companion model:
    $\boldsymbol{\omega}_{t+1} = \boldsymbol{\omega}_t - \eta_{\boldsymbol{\omega}} \nabla_{\boldsymbol{\omega}} [\mathit{\Delta}(f(\boldsymbol{\omega}), \alpha f(\boldsymbol{\omega}_t) + (1-\alpha) f(\boldsymbol{\theta}_t))]$
end for
Return : $\boldsymbol{\theta}_T$
```

Other Learning Settings. In addition to supervised learning, DCL can also be applied to other settings. For instance, we can employ DCL to fine tune a

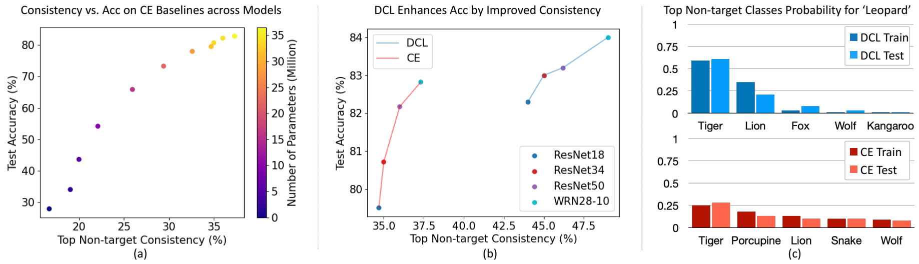

Fig. 2: Higher Top Non-target Consistency Indicates Better Generalization. We visualize training top non-target consistency and test accuracy with CIFAR-100 across different models. (a) Larger CE models have better generalization while having larger top non-target consistency. This indicates a positive correlation between top non-target consistency and test accuracy. (b) Across different architectures we see a consistent correlation of improved DCL test accuracy with increasing top non-target consistency. (c) DCL chooses Tiger and Lion most frequently as the top non-target classes of Leopard during training while CE exhibits inconsistent patterns.

pre-trained model on different downstream tasks. This is possible because the companion model does not use ground-truth labels for training, and thus can leverage unlabelled datasets. This naturally leads to a semi-supervised learning setting. DCL can also be employed for self-supervised pre-training. For example, we can employ DCL for Masked Autoencoder (MAE) [16] training. Here in lieu of cross-entropy loss used in classification, we replace the loss ℓ in Equation 1 to reconstruction loss. Similarly for the regularizer, we enforce consistency for the reconstructed output images. DCL can also be employed for Knowledge Distillation (KD) by adding an additional regularizer to the student during training. Intutitively, the student then seeks a consensus with both the teacher and the companion model.

3.4 Intuitive Justification

Consistency of top non-targets in training is correlated with improved generalization. We highlight salient aspects of DCL using experiments on CIFAR-100 across various architectures in Figure 2. The training data consists of N samples of K classes with N_c data samples for each class c. For data $(\mathbf{x}, y)$, we define the predicted logits from the model $\boldsymbol{\theta}$ as $f(\boldsymbol{\theta}, \mathbf{x}) \in \mathbb{R}^K$, where the k-th digit $f(\boldsymbol{\theta}, \mathbf{x})_k$ is the predicted probability of the k-th class. Then we define the top non-target class for the specific data as $\bar{c}_{\mathbf{x}} = \arg\max_{k, k \neq \mathbf{y}} f(\boldsymbol{\theta}, \mathbf{x})_k$, namely the class with the largest logits value among all the non-target classes. For a specific class c, its top non-target class consistency is defined as $\max_{k, k \neq c} \frac{\sum_{\mathbf{x}, y=c} \mathbb{1}(\bar{c}_{\mathbf{x}}=k)}{N_c}$, the percentage of the most frequent occurring top non-target class. For instance, Figure 2 (c) demonstrates for the class Leopard, the top non-target class is the Tiger with around 60% probability while the class Lion is 30% for both training

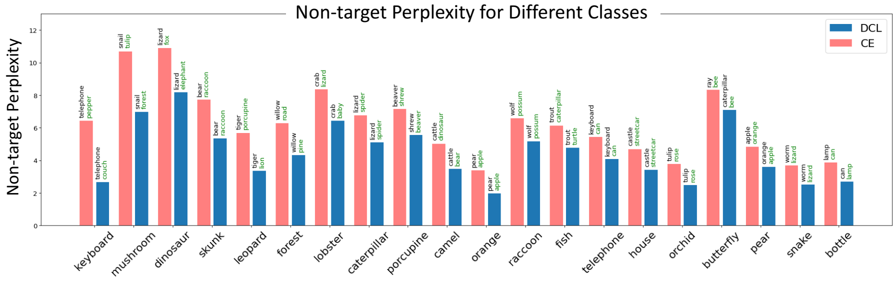

Fig. 3: Non-target Perplexity for Different Classes. The most (black text) and second most (green text) frequent classes as the top non-target class are shown on each bar. DCL can reduce perplexity over CE baseline.

and testing. This metric reflects how consistent the top non-target class is among the data of the same class. DCL effectively narrows the choices among the top non-target classes to tiger and lion while CE baseline evidently fails to capture this fine-grained semantic structure, with the probability of tiger or lion bearing similarity to other less meaningful classes. Therefore, the training consistency reveals the underlying semantic structure of the dataset in the logits space, which generalizes to test data as well. As expected, Figure 2 (a) depicts a positive correlation between training consistency and testing accuracy and Figure 2 (b) shows DCL can improve training consistency leading to improved test accuracy across diverse architectures.

Perplexity of non-targets class is lower for DCL compared with CE. In addition to examining the top non-target class, we also analyze the behavior of all non-target classes. We introduce non-target perplexity for each class, which is the perplexity over the conditional distribution of non-target classes occurring as the top non-target class, given a target class c. We define this conditional distribution $p(k|c) = \frac{\sum_{\mathbf{x},y=c} \mathbb{1}(\bar{c}_{\mathbf{x}}=k)}{N_c}, \forall k \in \{1,..,K\}, k \neq c$. We then define the perplexity $PP_c = \prod_{\forall k \in \{1,..,K\}, k \neq c} p(k|c)^{-p(k|c)}$. Lower perplexity indicates more consistency. Figure 3 compares the non-target perplexity between DCL and CE for 20 randomly selected classes in ResNet18 CIFAR100 experiments. Considering the class ’forest’ as an example, the two most frequent top non-target classes identified by CE are ’willow’ and ’road’, whereas for DCL, they are ’willow’ and ’pine’. ’Pine’ is semantically more reasonable than ’road’. As expected, DCL can reduce perplexity consistently over all classes.

4 Experiment

In this section, we evaluate the proposed DCL method on various datasets and architectures for different settings, and perform ablative studies.

Table 1: Performance Comparison on CIFAR-100, Tiny-ImageNet and ImageNet-1k. We benchmark DCL against CE and pre-trained baselines with various models. We report Gain as accuracy difference between DCL and CE. It clearly shows that DCL significantly outperforms CE methods. In addition, it reaches accuracy of ImageNet pre-trained(PT) baseline without any additional data and requires far less computation. (Standard errors for experiments on CIFAR-100 are within ±0.26, on ImageNet variants are within ±0.2).

CIFAR-100					**Tiny-ImageNet**				**ImageNet-1K**			
Model	**CE**	**PT**	**Ours**	**Gain**	**Model**	**CE**	**Ours**	**Gain**	**Model**	**CE**	**Ours**	**Gain**
ResNet18	78.3	80.5	82.4	**4.1**	ResNet18	61.8	64.6	**2.8**	ResNet18	69.5	70.5	**1.0**
ResNet34	80.6	82.6	83.1	**2.5**	ResNet34	64.0	66.8	**2.8**	ResNet50	76.0	77.1	**1.1**
ResNet50	82.1	82.9	83.6	**1.5**	ResNet50	64.4	67.0	**2.6**	ViT-T	65.7	66.5	**0.8**
ShuffeV2	72.8	73.5	74.0	**1.2**	ShuffeV2	53.9	55.5	**1.6**	Swin-T	81.3	81.7	**0.4**

4.1 Experimental Setup

Datasets. We consider publicly available image classification datasets: (a) CIFAR-100 [25] consists of 50K training and 10K test images from 100 classes with size $32 \times 32 \times 3$, (b) Tiny-Imagenet [28] contains 100K training and 10K test images from 200 classes with size $64 \times 64 \times 3$, and (c) ImageNet-1K [35] consists of 1.2M training and 100K test images from 1000 classes with size $224 \times 224 \times 3$. (d) For fine-tunning and and pre-trained downstream task, we also utilize CUB [41], Oxford-Pet [34], Food-101 [2] and Stanford Car [24] datasets. Same image sizes are applied for the fine-tunning datasets with the pre-trained dataset.

Settings. We apply DCL to supervised training on classification from scratch. Later on in the application section, we also utilize DCL for fine-tunning, semi-supervised learning, self-supervised pre-training and knowledge distillation tasks.

Baselines. We mainly compare DCL with standard cross-entropy baseline (CE). We also compare DCL with recent works with different optimization and regularization techniques [12, 13, 45]. Additionally, related baselines for self-distillation are also included [23, 46].

Models. We evaluate ResNet [15], ShuffleNetV2 [31], ViT [11] and Swin Transformer [29] architectures on these datasets. In particular, we benchmark ResNet18, ResNet34, ResNet50, and ShuffleNetV2 models on CIFAR-100 and Tiny-Imagenet dataset. Due to computing limitations, we only train ResNet18, ResNet50, Vit-t, Swin-t on ImageNet-1k dataset. For fine-tunning dataset, we benchmark EfficientNet [39] pre-trained on ImageNet. We provide their architectural details in supplementary.

Hyper-parameters. For the CIFAR-100 and Tiny-Imagenet datasets, we use the SGD optimizer with a momentum of 0.9 and weight decay of $5e-4$. We train these models up to 200 epochs with cosine learning rate decay with 0.1 as the initial learning rate and batch size of 128. For ImageNet-1k experiments, due to hardware limitations, we follow [45] setting using batch size of 256, 0.1 as the initial learning rate with cosine decay. We train ResNet for 90 epochs and ViTs for 300 epochs. For ResNet, we use an SGD optimizer with 0.9 as momentum and for the ViTs experiments, we use AdamW optimizer with $\beta_1 = 0.9$, $\beta_1 = 0.999$. All experiments use base augmentation on data. We provide the remaining hyper-parameter and experiments with different augmentations in supplementary.

4.2 Results

Table 1 compares the performance of our method and standard baseline on CIFAR-100, Tiny-Imagenet and ImageNet-1k datasets. Table 2 compares the performance of DCL with recent baselines for different optimization techniques including self-distillation. We report the accuracy of the final iteration in all the methods. Below, we highlight the main takeaway points.

DCL Achieves Better Generalization. We note DCL consistent improves upon CE across different datasets and architectures in Table 1 compared to baselines. For instance, on CIFAR-100 with ResNet18 architecture, CE achieves 78.3% accuracy while DCL achieves 82.4% accuracy. Experiments on Tiny-ImageNet shows a similar pattern. Table 2 shows DCL uniformly outperforms recent state-of-the-art methods across different backbones for the batch size and basic augmentation in [48] (Supplementary reports other augmentations).

Scalability to Large ImageNet-1k dataset. Table 1 shows that DCL scales well to large datasets such as Imagenet-1K. In particular, it achieves better accuracy than the baseline. For instance, with the ResNet50 architecture, the CE method achieves 76.0% accuracy while DCL achieves 77.1% accuracy.

Scalability to Transformer based architecture. Table 1 shows that DCL scales well to different tranformer-based backbones such as ViT and Swin transformer. For instance, with the ViT-T architecture, the CE method achieves 65.7% accuracy while DCL achieves 66.5% accuracy.

DCL Trained-from-scratch is Superior to ImageNet Pre-Trained Models. Table 1 shows the performance of the different models pre-trained on ImageNet and fine-tuned on the CIFAR-100 dataset. It takes CIFAR-100 $32 \times 32 \times 3$ image and scales to $224 \times 224 \times 3$ image and runs the inference using this input. In contrast, DCL trained the model with DCL using only CIFAR-100 data with $32 \times 32 \times 3$ input. DCL achieves better performance than the pre-trained counterpart. This is important because pre-training is expensive. For example when using a ResNet50 backbone, during inference DCL trained model requires 1298M MACs which is much lower than the 4198M MACs required by the ImageNet pre-trained model. In addition, the proposed method requires less data to achieve this performance, i.e., only 100K CIFAR-100 images compared to 1.2M ImageNet images. Thus, DCL yields faster inference and requires less sample complexity to achieve competitive performance as ImageNet pre-trained model.

Small DCL Models Outperform Large CE models. DCL trained on small models compares favorably with CE-trained large models. For instance, on CIFAR-100, ResNet18 trained with DCL achieves better accuracy than the much larger ResNet34 model trained with the cross-entropy method. Similar trend is evident in the context of ResNet34 vs. ResNet-50 performance. Furthermore, DCL trained

Table 2: Recent Baselines. DCL outperforms other baselines on various backones and datasets. Results reported are for the setup of GAM [45], which uses basic augmentations and 256 batch size for all ImageNet experiments. SAF [12] results on ImageNet use a batch size of 4096, an important factor for improved performance in ImageNet. Not reported here are more sophisticated augmentations, such as TrivialAugment(TA) [32]. In supplementary we show TA with CE gets 84.3%, consistent with that reported in [32]'s while TA with DCL 85.7% on CIFAR100 for WRN28-10. For ImageNet [32] uses a batch size of 2048, significantly larger than ours.

Methods	**CIFAR-100**		**ImageNet-1k**
	ResNet18	**WRN28-10**	**ResNet50**
CE [45]	78.3 ± 0.32	81.4 ± 0.13	76.0 ± 0.19
DML [46]	79.9 ± 0.32	82.7 ± 0.14	75.8 ± 0.15
SAM [12]	79.3 ± 0.25	83.4 ± 0.06	76.5 ± 0.11
PSKD [23]	80.6 ± 0.26	81.9 ± 0.10	76.3 ± 0.15
SAF [12]	80.8 ± 0.08	83.8 ± 0.04	76.4 ± 0.15
GAM [45]	80.5 ± 0.24	83.5 ± 0.09	76.6 ± 0.19
Ours (MSE)	$\mathbf{82.4 \pm 0.26}$	$\mathbf{84.2 \pm 0.10}$	$\mathbf{77.1 \pm 0.15}$

Table 3: Fine-tuning. Performance comparison with fine-tuning ImageNet pretrained EfficientNet. DCL outperforms cross-entropy on different downstream tasks.

Fine-tunning Methods	**Food-101**	**CUB-200**	**Oxford Pet**	**Stanford Car**
CE	82.5	63.8	90.9	78.2
DCL	**85.0**	**64.7**	**91.7**	**79.0**

ResNet34 gets better performance than the larger pre-trained ResNet50 model, showing further benefits of DCL.

4.3 Extended Applications

We extend DCL to different settings. We provide several examples to show the effectiveness of our method.

Fine-tuning. Table 3 shows the generalization of models when trained on sufficient labeled data and finetuned on a small dataset. We use ImageNet-1k pretrained EfficientNet as initial model and fine-tune on different smaller datasets. We outperform standard cross-entropy loss baselines. For example DCL achieves 91.7% accuracy compared with the baseline 90.9% on Oxford-Pet dataset.

Semi-supervised Learning. DCL can be applied on top of existing semi-supervised learning. While several methods, such as FixMatch [37], propose a consistency concept based on various input augmentations, DCL is focused on achieving consistency across different models realized along the training trajectory. This unique perspective allows DCL to be seamlessly integrated with existing semi-supervised learning techniques and leads to performance improvement.

We simply add our regularizer to their loss function and update $\boldsymbol{\theta}$. This is possible because the companion model does not require labels. Figure 4 shows comparison of FixMatch with and without DCL. With DCL, FixMatch gains 1% to 3% accuracy on CIFAR-10 classification with different number of labels. In particular, scenarios we gain more for fewer labels. For example, with only 40 labeled data, DCL can improve performance of FixMatch from 86.2% to 88.7%. All experiments are conducted with WRN-28-2 backbone with the same hyperparameters.

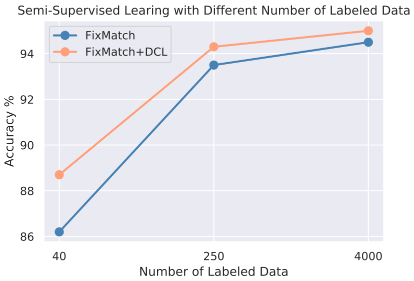

Fig. 4: Semi-Supervised Learning. DCL demonstrates superior performance over FixMatch, particularly with fewer labels.

Self-Supervised Pretraining. Instead of standard cross-entropy loss for classification task, self-supervised learning often focuses on different pretext tasks for pre-training. For example, the Masked Autoencoder (MAE) [16] is a variant of self-supervised learning that learns to predict or reconstruct the original images from partially masked or corrupted images. We replace the loss ℓ in Equation 1 with the reconstruction loss. For the regularizer, we enforce consistency with respect to reconstructed images. Due to computational limitation, we only finetune the pretrained model on classification tasks on several small datasets. Table 4 shows utilizing DCL to train MAE leads to better representations for downstream tasks. For example, DCL achieves 80% accuracy on CUB but vanilla MAE gets 79.2%. All experiments are conducted using ViT-Base backbone and we provide details for the hyperparameters in the supplementary.

Table 4: Self-Supervised Pretraining. Performance of MAE [16] on ViT-B backbone with and without DCL with fine-tunning downstream classification tasks. We use ImageNet-1k for pre-training and we use Tiny-ImageNet and other smaller dataset for fine-tunning. MAE with DCL outperforms plain MAE, showing benefits of DCL extension on self-supervised pre-training framework.

Methods	**ImageNet**	**Food-101**	**CUB-200**	**Oxford Pet**	**Stanford Car**
MAE [16]	82.8	87.8	79.2	91.5	82.5
MAE+DCL	**83.4**	**88.8**	**80.0**	**92.0**	**87.2**

Knowledge Distillation (KD). Knowledge Distillation similarly employs the concept of aligning the output distribution of two models, but alignment is achieved by the interaction between a smaller student network and a larger pre-trained teacher network. We apply DCL in addition to different KD methods in Table 5. DCL with student get 74.5% better than basic KD [19] without even

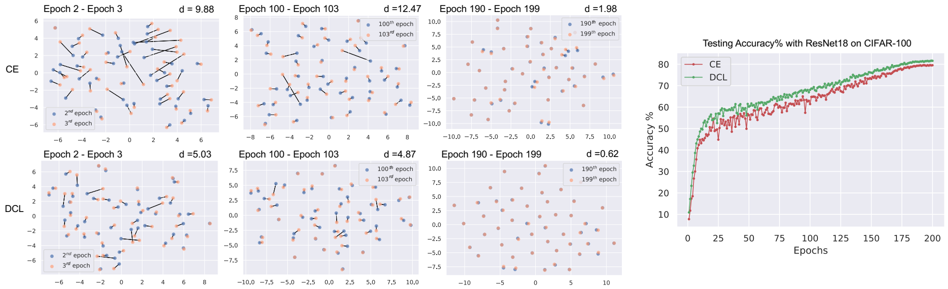

Fig. 5: Comparison of Model Variation and Test Accuracy along Training Trajectory on CIFAR-100 with ResNet18. The left section displays t-SNE visualizations of output logits for 50 test data samples, illustrating reduced variation in logits output across various training stages using DCL. In the plot, d is the average distance over data representation in the logit space. The right section presents the progression of test accuracy during training, DCL predictions show smaller variation and attain better generalization.

employing the large teacher model. DCL can also assist other SOTA methods [5,6,47] achieving enhanced performance.

Table 5: Knowledge Distillation. We shows results of ResNet8×4 as student and trained ResNet32×4 teacher model. DCL without trained teacher network outperforms KD [19]. DCL can improve other SOTA knowledge distillation methods [5,6,47].

	Teacher	**Student**	**KD [19]**	**ReviewKD [6]**	**SimKD [5]**	**DKD [47]**
w/o DCL	79.4	72.5	74.0	75.6	77.8	75.9
w/ DCL	-	**74.5**	**75.2**	**76.2**	**78.0**	**76.5**

DCL Reduces Model Variation and Improves Generalization. Figure 5 shows t-SNE visualization of logit space for test data to compare model variation and test accuracy along training ResNet18 on CIFAR-100 dataset. The figures shows changes of each data logits between neighboring epochs. At different stages of training, DCL consistently demonstrates reduced fluctuation in the logit space. This pattern is mirrored in a smoother accuracy trajectory during training, with DCL exhibiting superior performance compared to the CE baseline.

4.4 Ablations

DCL Generates Better Logit Space Representation. Figure 6 shows t-SNE visualization of logit space for 10 random classes of CIFAR-100 test data. DCL induced logit space enforces better linear separability of different classes.

Different Distance Functions. Instead of Mean Square Error as the distance function Δ, we can also use other forms of distance. Table 6 shows the benefit

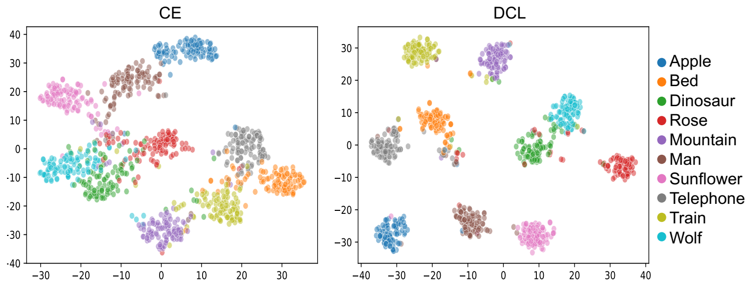

Fig. 6: Logits Visualization. t-SNE visualization of logit space from trained ResNet18 on CIFAR-100 using test data from 10 random classes.

Table 6: Ablative Study on Different Distance Functions. On ResNet-18 CIFAR-100 experiments, all variants of DCL are better than baseline while DCL with MSE as distance function gains the most.

Baseline	KL-Divergence	InfoNCE	L1	MSE
79.6	81.5	80.5	80.5	**82.4**

of DCL over all forms of distance functions Δ while MSE distance outperforms others reaching 82.4% on classification of CIFAR-100 with ResNet18 backbone.

5 Conclusion

We presented Deep Companion Learning (DCL), a novel DNN training approach that not only penalizes inconsistencies in model predictions but also leverages historical data to enhance generalization. Our strategy, centered around a deep-companion model (DCM), makes use of past predictions to enforce consistency and infer a meaningful semantic structure from the data. This novel mechanism introduces a dynamic, data-dependent regularization, optimizing both model consistency and the quality of representation in the logit space for improved class separability. Intuitively, DCL improves generalization during training by inferring a semantic structure for each class, and presenting the consistently reinforcing the confusing cases to the model. Our contributions include an Efficient Consistency Predictor that utilizes the companion model to dynamically adapt to new data, effectively minimizing prediction deviations. The empirical validation of DCL across various datasets and architectures demonstrates its capability to achieve state-of-the-art performance. Notably, it achieves comparable accuracy to models trained with pre-training while reducing computational demands. This highlights DCL's efficiency in training deep learning models from scratch. Furthermore, DCL is complementary to other settings including masked auto-encoders, fine-tuning, semi-supervised learning as well other methods using different augmentations.

Acknowledgments

We acknowledge the support by the Air Force Research Laboratory grant FA8650-22-C1039, Army Research Office grant W911NF2110246, and the National Science Foundation grants CCF-2007350 and CCF-1955981 and CPS Medium 2317079.

References

1. Ba, J.L., Kiros, J.R., Hinton, G.E.: Layer normalization. arXiv preprint arXiv:1607.06450 (2016)
2. Bossard, L., Guillaumin, M., Van Gool, L.: Food-101 – mining discriminative components with random forests. In: ECCV (2014)
3. Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., Joulin, A.: Emerging properties in self-supervised vision transformers. In: Proceedings of the International Conference on Computer Vision (ICCV) (2021)
4. Cha, J., Chun, S., Lee, K., Cho, H.C., Park, S., Lee, Y., Park, S.: Swad: Domain generalization by seeking flat minima. NeurIPS (2021)
5. Chen, D., Mei, J.P., Zhang, H., Wang, C., Feng, Y., Chen, C.: Knowledge distillation with the reused teacher classifier. In: CVPR (2022)
6. Chen, P., Liu, S., Zhao, H., Jia, J.: Distilling knowledge via knowledge review. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 5008–5017 (2021)
7. Chen, X., Fan, H., Girshick, R., He, K.: Improved baselines with momentum contrastive learning. arXiv preprint arXiv:2003.04297 (2020)
8. Cubuk, E.D., Zoph, B., Mane, D., Vasudevan, V., Le, Q.V.: Autoaugment: Learning augmentation policies from data. arXiv preprint arXiv:1805.09501 (2018)
9. Defazio, A., Bottou, L.: On the ineffectiveness of variance reduced optimization for deep learning. In: NeurIPS (2019)
10. DeVries, T., Taylor, G.W.: Improved regularization of convolutional neural networks with cutout. arXiv preprint arXiv:1708.04552 (2017)
11. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al.: An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929 (2020)
12. Du, J., Zhou, D., Feng, J., Tan, V., Zhou, J.T.: Sharpness-aware training for free. NeurIPS (2022)
13. Foret, P., Kleiner, A., Mobahi, H., Neyshabur, B.: Sharpness-aware minimization for efficiently improving generalization. In: ICLR (2021)
14. Grill, J.B., Strub, F., Altché, F., Tallec, C., Richemond, P., Buchatskaya, E., Doersch, C., Avila Pires, B., Guo, Z., Gheshlaghi Azar, M., Piot, B., kavukcuoglu, k., Munos, R., Valko, M.: Bootstrap your own latent - a new approach to self-supervised learning. In: NeurIPS (2020)
15. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016)
16. He, K., Chen, X., Xie, S., Li, Y., Dollár, P., Girshick, R.: Masked autoencoders are scalable vision learners. In: CVPR (2022)
17. He, K., Fan, H., Wu, Y., Xie, S., Girshick, R.: Momentum contrast for unsupervised visual representation learning. In: CVPR (2020)

18. Hendrycks, D., Mu, N., Cubuk, E.D., Zoph, B., Gilmer, J., Lakshminarayanan, B.: Augmix: A simple data processing method to improve robustness and uncertainty. arXiv preprint arXiv:1912.02781 (2019)
19. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531 (2015)
20. Hochreiter, S., Schmidhuber, J.: Flat minima. Neural computation **9**(1), 1–42 (1997)
21. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: ICML (2015)
22. Izmailov, P., Podoprikhin, D., Garipov, T., Vetrov, D., Wilson, A.G.: Averaging weights leads to wider optima and better generalization. arXiv preprint arXiv:1803.05407 (2018)
23. Kim, K., Ji, B., Yoon, D., Hwang, S.: Self-knowledge distillation with progressive refinement of targets. In: ICCV (2021)
24. Krause, J., Stark, M., Deng, J., Fei-Fei, L.: 3d object representations for fine-grained categorization. In: Proceedings of the IEEE international conference on computer vision workshops (2013)
25. Krizhevsky, A.: Learning multiple layers of features from tiny images (2009)
26. Krogh, A., Hertz, J.: A simple weight decay can improve generalization. NeurIPS (1991)
27. Laine, S., Aila, T.: Temporal ensembling for semi-supervised learning. In: ICLR (2017)
28. Le, Y., Yang, X.: Tiny imagenet visual recognition challenge (2015)
29. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B.: Swin transformer: Hierarchical vision transformer using shifted windows. In: ICCV (2021)
30. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101 (2017)
31. Ma, N., Zhang, X., Zheng, H.T., Sun, J.: Shufflenet v2: Practical guidelines for efficient cnn architecture design. In: ECCV (2018)
32. Müller, S.G., Hutter, F.: Trivialaugment: Tuning-free yet state-of-the-art data augmentation. In: ICCV (2021)
33. Neu, G., Dziugaite, G.K., Haghifam, M., Roy, D.M.: Information-theoretic generalization bounds for stochastic gradient descent. In: COLT (2021)
34. Parkhi, O.M., Vedaldi, A., Zisserman, A., Jawahar, C.: Cats and dogs. In: CVPR (2012)
35. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al.: Imagenet large scale visual recognition challenge. IJCV **115**(3), 211–252 (2015)
36. Shalev-Shwartz, S., Ben-David, S.: Understanding Machine Learning - From Theory to Algorithms. Cambridge University Press (2014)
37. Sohn, K., Berthelot, D., Carlini, N., Zhang, Z., Zhang, H., Raffel, C.A., Cubuk, E.D., Kurakin, A., Li, C.L.: Fixmatch: Simplifying semi-supervised learning with consistency and confidence. In: NeurIPS (2020)
38. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research (2014)
39. Tan, M., Le, Q.: Efficientnet: Rethinking model scaling for convolutional neural networks. In: ICML (2019)
40. Tarvainen, A., Valpola, H.: Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In: NeurIPS (2017)

41. Wah, C., Branson, S., Welinder, P., Perona, P., Belongie, S.: The caltech-ucsd birds-200-2011 dataset (2011)
42. Wu, Y., He, K.: Group normalization. In: ECCV (2018)
43. Yun, S., Han, D., Oh, S.J., Chun, S., Choe, J., Yoo, Y.: Cutmix: Regularization strategy to train strong classifiers with localizable features. In: ICCV (2019)
44. Zhang, H., Cisse, M., Dauphin, Y.N., Lopez-Paz, D.: mixup: Beyond empirical risk minimization. arXiv preprint arXiv:1710.09412 (2017)
45. Zhang, X., Xu, R., Yu, H., Zou, H., Cui, P.: Gradient norm aware minimization seeks first-order flatness and improves generalization. In: CVPR (2023)
46. Zhang, Y., Xiang, T., Hospedales, T.M., Lu, H.: Deep mutual learning. In: CVPR (2018)
47. Zhao, B., Cui, Q., Song, R., Qiu, Y., Liang, J.: Decoupled knowledge distillation. In: Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition (2022)
48. Zilly, J.G., Srivastava, R.K., Koutník, J., Schmidhuber, J.: Recurrent highway networks. In: ICML. pp. 4189–4198 (2017)