Nearest Class-Center Simplification through Intermediate Layers

Ido Ben-Shaul [1 2] **Shai Dekel** [1]

Abstract

Recent advances in theoretical Deep Learning have introduced geometric properties that occur during training, past the Interpolation Threshold-where the training error reaches zero. We inquire into the phenomena coined *Neural Collapse* in the intermediate layers of the networks, and emphasize the innerworkings of Nearest Class-Center Mismatch inside the deepnet. We further show that these processes occur both in vision and language model architectures. Lastly, we propose a Stochastic Variability-Simplification Loss (SVSL) that encourages better geometrical features in intermediate layers, and improves both train metrics and generalization.

1. Introduction

Several recent works have investigated the nature of modern Deep Neural Networks (DNNs) past the point of zero training error (Belkin, 2021; Nakkiran et al., 2020; Bartlett et al., 2021; Power et al., 2022). The stage at which the training error reaches zero is called the *Interpolation Threshold (IT)*, since at this point, the learned network function interpolates between training samples. This is not to be confused with zero-loss, but simply the point where all training samples are correctly classified. The stage of training beyond the IT is coined the *Terminal Phase of Training (TPT)* in (Papyan et al., 2020). It was in this paper that the term *Neural Collapse (NC)* was introduced to describe four interconnected geometrical phenomena that describe the network behavior past the TPT. Let us briefly describe the properties of NC that are most relevant for this paper:

(NC1) Variability collapse: As training progresses, the within-class variation of the activations becomes negligible as these activations collapse to their class-means.

(NC4) Simplification to Nearest Class-Center (NCC): For a given deepnet activation, the network classifier converges to choosing whichever class has the nearest train class-mean (in standard Euclidean distance).

In this work we delve deeper into the inner workings of NCC Simplification. Several works have proposed that geometrical properties of intermediate layers shape the way deepnets are trained, and largely affect their successes (Cohen et al., 2020; Allen-Zhu & Li, 2020; Ben-Shaul & Dekel, 2021; Liu & Arik, 2020; Alain & Bengio, 2017; Baldock et al., 2021). We explore along the lines of (Papyan et al., 2020) to further understand such geometries.

1.1. Our contributions

Our contributions can be summarized as follows:

NCC Simplification in Intermediate Model Layers: We show that when looking at the NCC Mismatch of deepnet intermediate layers, before and during TPT, there is a beautiful geometric structure that emerges. Namely:

(i) There is a clear ordering between NCC mismatch in intermediate layers. The mismatch is lower as the layers gets deeper.

(ii) NCC Simplification is not only apparent in the final layer of the network, and may propagate back several layers in the network.

NCC-Simplification is apparent in in Transformer NLP architectures: When proposed in (Papyan et al., 2020), the authors show that Neural Collapse appears in several well-known Image Classification models: VGG (Simonyan & Zisserman, 2015), ResNet (He et al., 2016), and DenseNet (Huang et al., 2017) on Image Classification datasets. In this paper, we show that NCC simplification is also apparent in Transformer architectures (Vaswani et al., 2017), and even more surprisingly, in common NLP tasks. The recent surge in Transformer architectures in cross-modal tasks, suggests that there are common behaviors between classic Image architectures, and more recent mechanisms (Radford et al., 2021; Dai et al., 2021; Raghu et al., 2021).

NCC-Simplification guiding can improve training and generalization: We propose a simple intermediate layer variance collapsing loss which we coin the Stochastic Variability Simplification Loss (SVSL). This loss is shown to

[1]Department of Applied Mathematics, Tel-Aviv University [2]eBay Research. Correspondence to: Ido Ben-Shaul <ido.benshaul@gmail.com>.

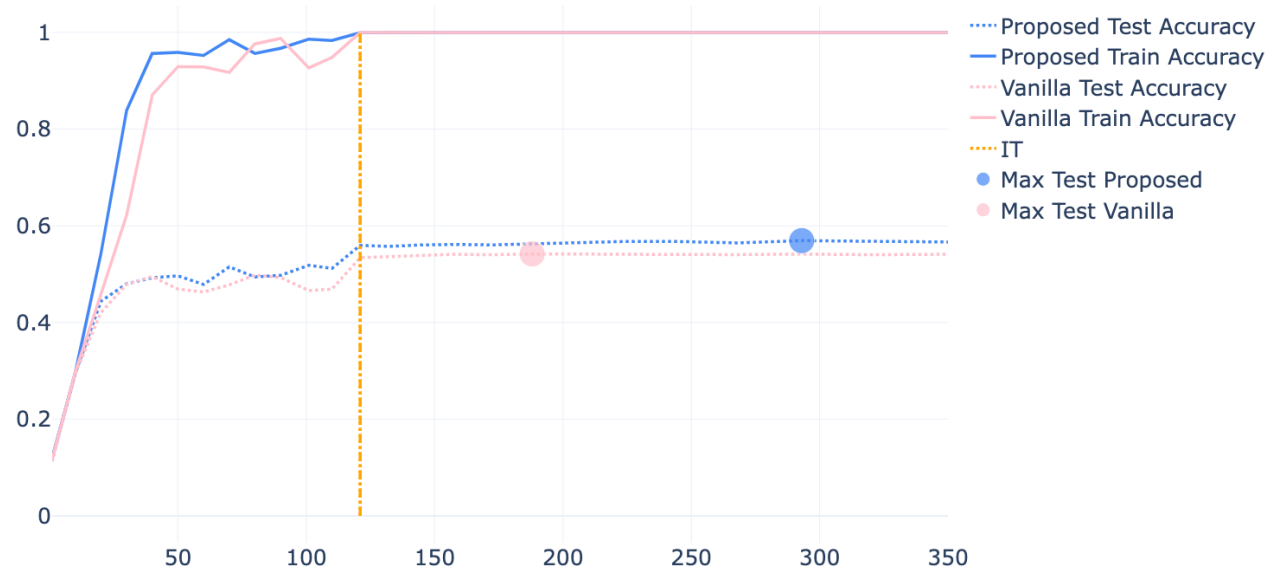

Figure 1. **STL-10** experiment training procedure. We show the accuracy on both the train and test set, using the different losses: Vanilla Cross-Entropy (**Pink**) and SVSL (**Blue**). The SVSL loss outperforms the vanilla in both metrics. Both models improve in terms of test-performance during the TPT - the stage past the IT (**Orange**). In the TPT, both models have a constant accuracy of 1.

improve the performance of a wide-variety of tasks by encouraging NCC simplification during training. We show that this loss is able to improve NCC mismatch between intermediate layers, on both the train and test datasets. The different training stages and their respective metrics can be seen for both losses on the STL-10 dataset in Figure 1. The same plot is given for all other datasets in a later part of the paper. In all plots in the paper, the x-axis represents the epochs during train. We share the code for reproducing the paper experiments in the supplementary materials.

2. Problem Setup and Background

2.1. Supervised Classification

We are interested in the supervised classification setting. In this paper, our experiments include problems in both Image and NLP-Sequence Classification. In **Image Classification** we are given a training set of $d := 3 \times W \times H$-dimensional RGB images, from C categories. We wish to train a network to differentiate between different image classes. On the other hand, in **Text-Sequence Classification** we are given a training set of sequences. We use standard tokenization techniques to transfer discrete sequences to a continuous euclidean space. Since different sequences may be in different lengths, we pad sequences in our experiments to a certain $d := \text{MAX\_PAD}$ constant. Similar to the Image setting, we wish to find the appropriate class for each text-sequence(of dimension $\mathbb{R}^d$, for C ground-truth classes).

Let g represent a deepnet, $g : \mathbb{R}^d \to \mathbb{R}^C$ where C is the number or classes, and network parameters θ which are learned through the optimization procedure, or "training". For the classification setting, the classification decision of the network for input $x \in \mathbb{R}^d$ is defined as $z = \arg\max_{1 \leq c' \leq C} g(x)_{c'}$. Let $L = \{l^{(1)}, \ldots, l^{(k)}\}$ represent the set of intermediate layers of network g (formally defined in Section 5.3), such that $g := l^{(k)} \circ \cdots \circ l^{(1)}$. Let us also define n_j as the output dimension of layer j, such that $n_0 := d$ and $n_k := C$. We note $g^{(j)}(x)$ as the outputs of the j^{th} layer of the network for input sample x. Using these definitions, the following holds: $g^{(j)}(x) := l^{(j)} \circ \cdots \circ l^{(1)}(x)$.

2.2. Representation Learning

In recent years, the fields of Vision and NLP have both been transformed by representation learning methods in supervised and unsupervised tasks. The main premise is to learn "representations such that similar samples stay close to each other, while dissimilar ones are far apart" (Weng, 2021). Encouraging clustering in features learned by a deepnet has been a pivotal early idea to improve representations (Xie et al., 2016; Tian et al., 2017). In this paper we will show that encouraging clustering of intermediate layers can

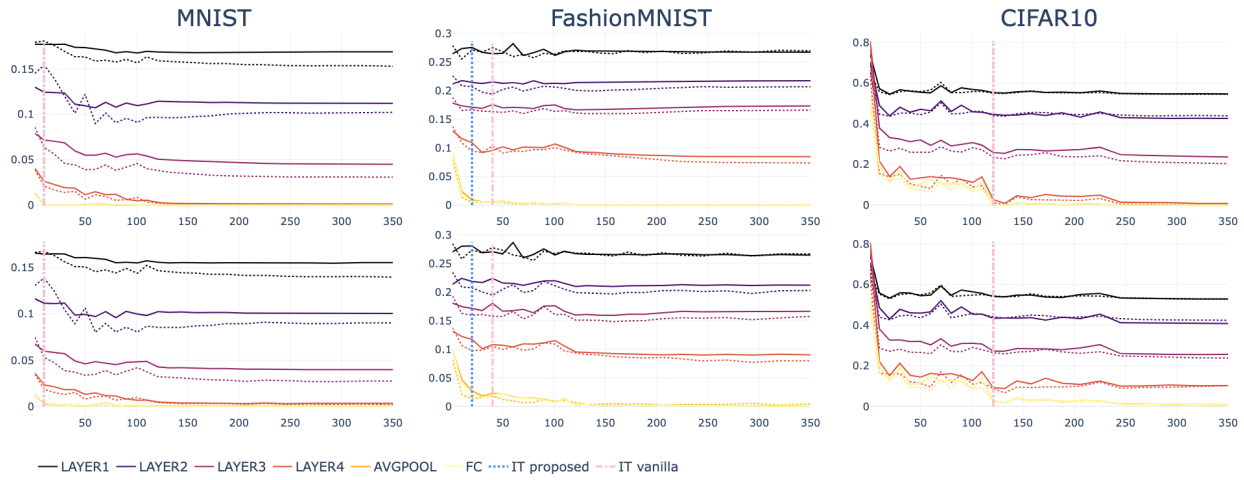

Figure 2. NCC Mismatch for Vision datasets: **MNIST**, **Fashion-MNIST** and **CIFAR10**, using both vanilla (**solid**) and SVSL (**dashed**) losses. **Top:** train NCC Mismatch, **Bottom:** test NCC Mismatch. The pink vertical line shows the IT for the vanilla loss, while the blue vertical line shows the IT for SVSL. In most cases the two IT intersect.

boost performance. We use the normal convention of calling a sample the *Anchor*, a similar example *Positive* and a dissimilar example as *Negative*. In Vision, representation learning can be approached by using clustering assignments as pseudo-labels (Tian et al., 2017), invoking similarity between different augmentation of the same sample (Zbontar et al., 2021; Chen et al., 2020; Caron et al., 2021), or even using class labels to wisely pick Positive and Negative items (Khosla et al., 2020). In Language Modeling(LM), Masking and Next-Sentence prediction(among others) tasks are used to learn semantically robust representations (Devlin et al., 2019; Liu et al., 2019). We use the notion of representation learning by enforcing low inner-class variability. The methods mentioned only use the final layer of the encoder to penalize the representations. We conjure that better consistency of class representations in intermediate layers forces representations in final layers to have better geometrical features. Our method does not need to sample pairs(positive or negative), and is simple to implement.

3. Neural Collapse

We shall now briefly present the particular properties of Neural Collapse that are relevant for this work, as presented in (Papyan et al., 2020; Han et al., 2021). Let g be a given network and $\{(x_i, y_i)\}_{i\in\mathcal{I}_{\text{Train}}}$, $\{(x_i, y_i)\}_{i\in\mathcal{I}_{\text{Test}}}$ be the train and test set accordingly.

Definition 1 (Train class-means)**.** *We define the train class means for layer $l^{(j)}$ and class $1 \le c \le C$ as*

$$\mu_c^{(j)} := \text{Avg}_{i\in\mathcal{I}_{\text{Train}}, y_i=c}\{g_i^{(j)}\}.$$

Definition 2 (Train within-class covariance)**.** *We define the train within-class covariance for layer $l^{(j)}$ as*

$$\Sigma_W^{(j)} := \frac{1}{C}\sum_{c=1}^{C} \text{Avg}_{i\in\mathcal{I}_{\text{Train}}, y_i=c}\{(g_i^{(j)}-\mu_c^{(j)})(g_i^{(j)}-\mu_c^{(j)})^\top\}.$$

Using these definitions, we can formally define properties (NC1) and (NC4) from (Papyan et al., 2020). Let us assume that the network g may be split into two stages: the "feature engineering" stage $g^{(k-1)}$ and the final classifier layer, $l^{(k)}$ such that $g := l^{(k)} \circ g^{(k-1)}$.

Definition 3 (**NC1** Variability Collapse)**.** $\Sigma_W^{(k-1)} \to 0$.

Definition 4 (**NC4** Simplification to NCC)**.** *Let:*

$$S := \Big\{ i \in \mathcal{I}_{\text{Train}} \mid \arg\max_{1\le c\le C} g\left(x_i\right)_c \neq \arg\min_{1\le c\le C} \left\| g^{(k-1)}\left(x_i\right) - \mu_c^{(k-1)} \right\|_2 \Big\},$$

then $|S| \to 0$, where $|X|$ is the number of elements in a finite set X.

Throughout this paper, we make the assumption that the deepnets are of proper capacity to reach the TPT, or in other words to "fit" the data. In the experiments we use large architectures that are highly overparameterized for the given tasks.

4. Contributions

4.1. NCC mismatch in Intermediate Layers

The results of (Papyan et al., 2020) show a clear behavior in terms of NCC-Simplification in the final layer (see Definition 4). We further investigate the behavior of intermediate layers in terms of NCC mismatch. Let us define:

Definition 5 (Layer j **Train** NCC mismatch)**.**

$$\Lambda_{\text{Train}}^{(j)} := \frac{1}{N_{\text{Train}}} \left| \left\{ \arg\max_{1\le c'\le C} g\left(x_i\right)_{c'} \neq \arg\min_{c'} \left\| g^{(j)}\left(x_i\right) - \mu_{c'}^{(j)} \right\|_2 \mid i \in \mathcal{I}_{\text{Train}} \right\} \right|. \tag{1}$$

Definition 6 (Layer j **Test** NCC mismatch)**.**

$$\Lambda_{\text{Test}}^{(j)} := \frac{1}{N_{\text{Test}}} \left| \left\{ \arg\max_{1\le c'\le C} g\left(x_i\right)_{c'} \neq \arg\min_{c'} \left\| g^{(j)}\left(x_i\right) - \mu_{c'}^{(j)} \right\|_2 \mid i \in \mathcal{I}_{\text{Test}} \right\} \right|. \tag{2}$$

Our conjectures can now be described as follows

Conjecture 7 (Intermediate Layer ordering using NCC mismatch)**.** *There is a clear order between both **train** and **test** NCC mismatch in intermediate layers. The mismatch is lower as the layers gets deeper. In the TPT, for $1 \le j \le k$,*

$$\Lambda_{\text{Train}}^{(j)} \ge \Lambda_{\text{Train}}^{(j+1)} \quad \text{and} \quad \Lambda_{\text{Test}}^{(j)} \ge \Lambda_{\text{Test}}^{(j+1)}. \tag{3}$$

Conjecture 8 (NCC mismatch improves in TPT)**.** *At each intermediate layer, both the **train** and **test** NCC mismatch improves from the IT to End of Training(EOT).*

$$\Lambda_{\text{Train,IT}}^{(j)} \ge \Lambda_{\text{Train,EOT}}^{(j)} \quad \text{and} \quad \Lambda_{\text{Test,IT}}^{(j)} \ge \Lambda_{\text{Test,EOT}}^{(j)}. \tag{4}$$

4.2. Decreasing NCC Mismatch using Stochastic Variability-Simplification Loss

In (Papyan et al., 2020), the properties shown in Definitions 3 and 4 both act on the final layer of the network. When considering intermediate feature spaces, the property shown in Definition 3 amounts to promoting class clustering. Promoting class clustering can push samples further from decision boundaries between classes, and could therefore increase agreement between the nearest class-center and the classifier. We wish to decrease the NCC mismatch during train, and encourage better clustering through the intermediate layers. Our loss function is proposed as follows:

Definition 9 (Stochastic Train class-means)**.** *Let $\mathcal{B} := \{(x_j, y_j)\}_{j\in B}$, where $|B|$ is the Batch-Size. We define the stochastic train class means for layer $l^{(j)}$, batch $\mathcal{B}$, and class $1 \le c \le C$ as*

$$\mu_{c,\mathcal{B}}^{(j)} := \text{Avg}_{i\in\mathcal{B}, y_i=c}\{g_i^{(j)}\}.$$

Definition 10 (Stochastic Variability-Simplification Loss (SVSL))**.** *Let g be a deepnet and $\hat{y}_i = g(x_i)$ for $(x_i, y_i), i \in \mathcal{I}_{\text{Train}}, y_i = c, 1 \le c \le C$. Let $\mathcal{B}$ be the batch such that $i \in B$. We also define $\gamma \in \mathbb{N}, 1 \le \gamma < k$ and $\alpha \in \mathbb{R}_+$ two hyperparameters. The Stochastic Variability-Simplification Loss function is then defined as*

$$\mathcal{L}\left(\hat{y}_i, y_i\right) := \text{CE}\left(\hat{y}_i, y_i\right) + \eta \sum_{j=\gamma}^{k} \left\| g^{(j)}\left(x_i\right) - \mu_{c,\mathcal{B}}^{(j)} \right\|_2^2. \tag{5}$$

where CE is the well-known Cross-Entropy loss and

$$\eta = \frac{\alpha}{C\left(k+1-\gamma\right)\left|\{i \in B \mid y_i = c\}\right|}.$$

It is also possible to define the Variability-Collapse in a non-stochastic fashion, by computing the full class-means at layer for every epoch. An example implementation of the SVSL is given in the appendix.

Using the SVSL, we claim the following behaviors:

Conjecture 11 (SVSL improves NCC mismatch)**.** *Using the properly defined hyperparameters α, γ, the Stochastic Variability-Simplification Loss encourages lower **train** and **test** NCC mismatch in intermediate layers. In the TPT, for $1 \le j \le k$,*

$$\Lambda_{\text{Train,Vanilla}}^{(j)} \ge \Lambda_{\text{Train,SVSL}}^{(j)} \quad \text{and} \quad \Lambda_{\text{Test,Vanilla}}^{(j)} \ge \Lambda_{\text{Test,SVSL}}^{(j)}. \tag{6}$$

Conjecture 12 (SVSL can improve test-performance)**.** *The EOT test metrics are improved for all datasets using the SVSL and proper hyperparameter tuning.*

4.2.1. Motivation for SVSL

The Folding Ball Hypothesis is presented in (Chollet, 2017) as follows: "Imagine two sheets of colored paper: one red and one blue. Put one on top of the other. Now crumple them together into a small ball. That crumpled paper ball is your input data, and each sheet of paper is a class of data in a classification problem. What a neural network is meant to do is figure out a transformation of the paper ball that would uncrumple it, so as to make the two classes cleanly separable again". This geometrical notion has been used to try to predict the wellness of such transformations, using their geometrical properties (Cohen et al., 2020; Ben-Shaul & Dekel, 2021; Alain & Bengio, 2017; Montavon et al., 2011). When measuring NCC mismatch during TPT, the network has near 0-training-error. This essentially means that the final feature space(where the inputs to the classifier reside) has near perfect clusters per-class. In (Papyan et al., 2020), it is empirically shown and proven that for most deepnets, the final layer has a single cluster for each class. Thus, measuring the **train** NCC mismatch between the jth

feature-space and the classifier is similar to checking the NCC mismatch with the ground-truth labels. The clustering of feature spaces is an iterative transformation from each layer to the next, where the quality of clustering assists in clustering at the following stage.

Demanding a low NCC mismatch in early layers of the network may be unsatisfiable, as the input samples (e.g. images) cannot necessarily be well clustered with such low capacity (small number of layers). This is the reason we allow the γ hyperparameter to facilitate the earliest layer from which we require the SVSL. Demanding consistency between the NCC and the classifier early in training can interfere with the model learning the proper class predictions, so we leverage between the losses using the α hyperparameter. A different approach can consist of applying the SVSL only during TPT.

5. Experiment Details

Our experiments aim to demonstrate Conjectures 7,8,11,12 on both Vision and NLP tasks. In Section 5.1 we introduce the datasets that were used. In Section 5.2 we describe the architectures and Section 5.3 goes through the training procedures used.

5.1. Datasets

For **Vision** tasks, we use most of the datatsets used in (Papyan et al., 2020). Namely: MNIST, FashionMNIST, CIFAR10, CIFAR100, and STL10. Unlike in (Papyan et al., 2020), we do not balance the datasets explicitly and keep them as they are. We use mean-std train normalization. Inorder to get intermediate features, we use PyTorch Hooks (Paszke et al., 2019). For the **NLP sequence classification tasks** we use a subset of binary datasets from the GLUE benchmark (Wang et al., 2018). We run our experiments on datasets from all three types of tasks: *Single-Sentence Tasks:* CoLA and SST-2, *Similarity and Paraphrase Tasks:* MRPC, and *Inference Tasks:* RTE. All datasets have 2 classes. In order to make all sequences of the same length, both for computing NCC mismatch and maintaining same size features, we pad each of the sequences in all datasets to 32 tokens. Intermediate features are readily given as "$\mathrm{hidden\_states}$" in (Wolf et al., 2020).

5.2. Architectures

Vision: For the vision architectures we follow the guidelines set in (Papyan et al., 2020). In this paper we use solely the ResNet (He et al., 2016) architectures. ResNet18 is used for MNIST, FashionMNIST, and CIFAR10. For CIFAR100 and STL10 the model chosen is the ResNet50 architecture. The layers for the ResNet architecture that are used in the experiments are $\{\mathrm{Layer1}, \mathrm{Layer2}, \mathrm{Layer3}, \mathrm{Layer4}, \mathrm{AvgPool}, \mathrm{FC}\}$ as implemented in TorchVision (Marcel & Rodriguez, 2010). **NLP Sequence Classification:** For all sequence classification task we use an Uncased pre-trained BERT (Devlin et al., 2019). The layers used for this architecture are the hidden states of the BERT-architecture. We include the embedding-layer in the BERT architecture, and use all hidden-states except the final output layer. In total, we have an initial embedding features (1) and (11) hidden-state layers, for a total of 12 layers in this architecture. In theory, the final layer can also be used in the optimization process.

5.3. Optimization Procedure

Vision: We use the same optimization scheme as in (Papyan et al., 2020), using best training hyperparameters as logged, and follow the same training procedure. We train all datasets for 350 Epochs. The Batch-Size for all experiments is 128. All vision experiments are trained using a SGD optimizer as done in the original paper. All SVSL Hyperparameters used are given in Table 3. We report the top-1-accuracy on the test datasets. We use a threshold of 0.995 for determining the Interpolation Threshold.

NLP Sequence Classification: We follow the default hyperparameters as shown in (Wolf et al., 2020) (GLUE finetune example). All experiments are trained using an AdamW (Loshchilov & Hutter, 2019) optimizer, and the default hyperparameters for 10 epochs. The Batch-Size for all experiments is 8, and the tasks are all binary classification. We report test-accuracy for the datasets: RTE, SST-2, and MRPC, and Matthew's-Correlation for the CoLA dataset. **SVSL Hyperparameters:** The SVSL parameters are found using a simple baysean optimization scheme (AXBoTorch (Balandat et al., 2020)) for $\alpha \in [5e-8, 5e-5]$ and layers $\gamma \in \{1, \ldots, 11\}$ on the test set. The purpose of these experiments is to show the ability to improve the network behavior using the SVSL. Possible future research includes adding the hyperparameters to as part of the network weights. The hyperparameters used are recorded in Table 3. We use a threshold of 0.985 for determining the IT.

6. Results

6.1. NCC mismatch Behavior in Intermediate Layers

We wish to demonstrate the Conjectures 7 and 8. Table 1 attains to both of these statements for the given vision datasets on the test-NCC mismatch. The train and test NCC mismatch metrics for the MNIST, Fashion-MNIST, and CIFAR10 datasets are visualized in Figure 2 (Solid Lines). The same metrics for the sequence classification tasks: are given in Figure 3. The results and visualizations for the remaining datasets are given in the appendix.

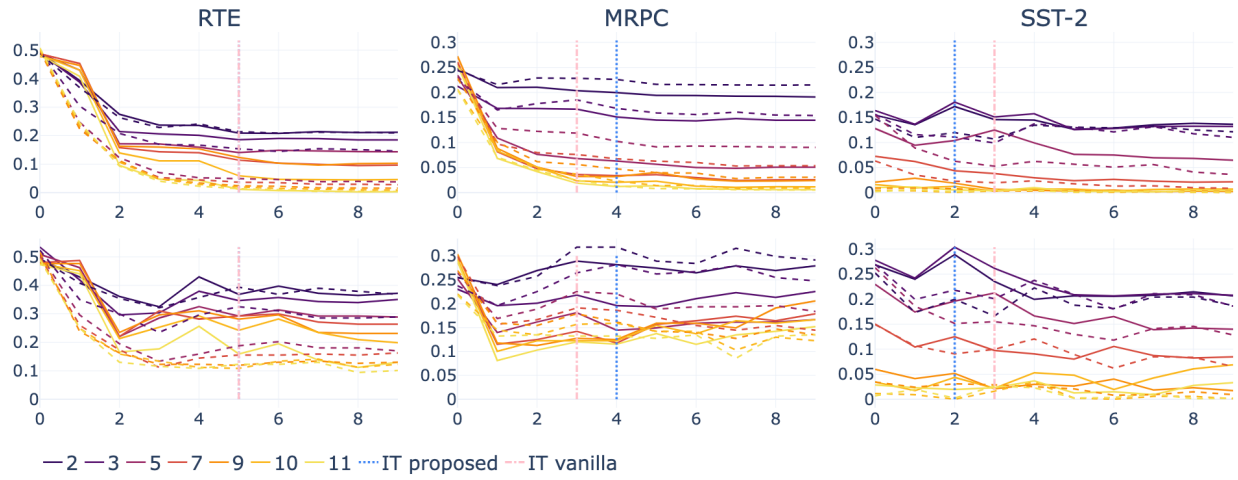

Figure 3. NCC mismatch for Sequence-Classification datasets: **RTE**, **MRPC** and **CoLA**, using both vanilla (**solid**) and SVSL(**dashed**) losses. **Top:** train NCC mismatch, **Bottom:** test NCC mismatch. We show only a subset of the transformer layers for clearness. The pink vertical line shows the IT for the vanilla loss, while the blue vertical line shows the IT for SVSL.

Table 1. Comparing the **test** NCC mismatch at IT vs. EOT for Image Classification Datasets using the vanilla model architecture(scores are in percents)

	Layer 1		**Layer 2**		**Layer 3**		**Layer 4**		**Avg. Pooling**		**FC**	
Dataset	**IT**	**EOT**	**IT**	**EOT**	**IT**	**EOT**	**IT**	**EOT**	**IT**	**EOT**	**IT**	**EOT**
MNIST	16.47	**15.57**	11.16	**10.05**	5.97	**3.97**	2.33	**0.34**	0.22	**0.01**	0.16	**0.01**
F-MNIST	26.7	**26.44**	**20.97**	**21.2**	**15.97**	16.62	9.52	**9.01**	0.3	**0.11**	0.3	**0.13**
STL10	**59.74**	**59.84**	57.62	**56.96**	52.72	**51.36**	20.42	**15.32**	**0.1**	0.4	**0.0**	**0.02**
CIFAR10	54.09	**52.88**	43.48	**40.8**	27.19	**25.59**	**9.17**	10.22	2.61	**0.37**	2.61	**0.37**
CIFAR100	77.19	**76.62**	70.61	**69.76**	60.02	**58.56**	43.76	**37.82**	18.76	**5.23**	18.76	**5.23**

6.2. Variability-Simplification Loss

In this section we wish to demonstrate how using the intermediate-layer SVSL can improve training procedure and generalization. In Section 4.2.1 we describe the underlying logic behind the proposed cost. We advocate that in networks where intermediate NCC mismatch is lower, perform better in the TPT stage. Let us first demonstrate the correctness of Conjecture 11. Figure 2 (Dashed Line) shows the **train** and **test** NCC mismatch of the network using the SVSL with the proposed hyperparameters, for MNIST, FashionMNIST, and CIFAR10. The visualization for the remaining Image datasets is given in the appendix. It is clear that for all datasets, and almost all layers, the NCC mismatch improves when using the SVSL. The same conclusions can be derived for all NLP datasets in Figure 3.

We shall show the validity of Conjecture 12. Table 2 compares the test-performance of the vanilla Cross-Entropy (CE) loss with that of the SVSL on all datasets. This comparison is done at the IT, EOT, and also at the best Test-Epoch. We see that SVSL outperforms the vanilla CE at almost all stages of training. We also see that most datasets reach their best Test-Scores during TPT. Even when using the best possible Test Epoch, the SVSL loss achieves better or as-good results in all but one dataset. When the best Test Epoch is not achieved in the TPT, the scores achieved at the best Test-Epoch are comparable to the ones achieved at EOT. Practitioners in the field often look at regions of near-zero training-error, and use a validation set to choose the proper early-stopping criterion. This stage is formally given as the TPT, and hence a convincing method is to look at the performance mainly in this region. In these tasks we use the testing set as a proxy for the validation set. We see that even when allowing ourselves to look at all Test-Scores, the SVSL still achieves better performance on an array of tasks. All training graphs with both losses are given for the Vision datasets in Figures 1, 4 and for the NLP datasets in 5. In practice, one may use a hold-out validation/cross-validation set to choose the best epoch and achieve similar results to the maximal points in the plots.

Table 2. Comparing the Test Metrics of CE-Loss(Vanilla) with Stochastic Variability-Simplification Loss (SVSL) at IT, EOT, and Best-Test-Epoch for both Image and NLP-Sequence Classification Datasets. The metrics are as defined in Section 5.3 in percents. The Matthew's -Correlation metric for the CoLA dataset is multiplied by a factor of 100. We also note whether the best Test-Metrics are achieved in TPT, for all datasets and methods.

	IT		**EOT**		**Best Test Epoch**			
Dataset	Vanilla	SVSL	Vanilla	SVSL	Vanilla	In TPT	SVSL	In TPT
MNIST	**99.37**	**99.36**	99.61	**99.69**	99.65	Yes	**99.69**	Yes
Fashion MNIST	91.78	**93.13**	93.82	**93.88**	93.93	Yes	**94.03**	Yes
STL10	53.41	**55.95**	54.11	**56.65**	54.19	Yes	**56.94**	Yes
CIFAR10	**80.64**	80.56	80.96	**81.19**	80.96	Yes	**81.19**	Yes
CIFAR100	52.77	**53.28**	53.31	**54.29**	53.79	Yes	**54.29**	Yes
CoLA	51.59	**52.91**	53.46	**55.54**	53.95	No	**55.54**	Yes
RTE	**58.84**	58.12	55.23	**59.57**	**61.01**	No	60.28	Yes
MRPC	70.83	**74.26**	74.26	**75.25**	75.00	No	**76.71**	No
SST-2	87.96	**88.42**	88.42	**88.76**	**89.22**	No	**89.22**	Yes

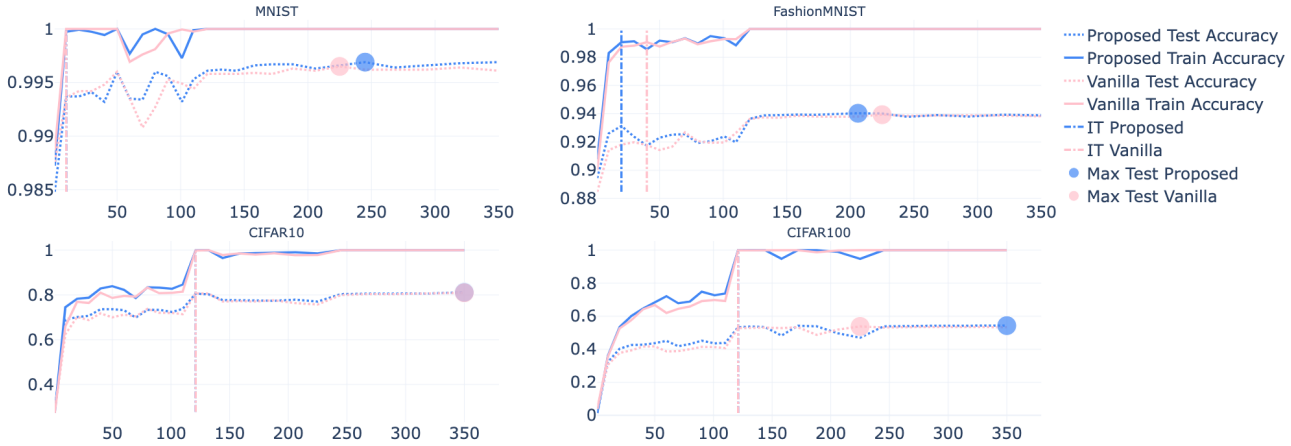

Figure 4. Optimization Procedures for Vision experiments: **MNIST**, **Fashion-MNIST**, **CIFAR10**, **CIFAR100**. The SVSL achieves higher performance(test and train) at most epochs. For the Vision datasets, all models achieve best test performance during the TPT.

Table 3. SVSL-Hyperparametrs used for **Image** and **NLP sequence-classification** tasks, optimized using BoTorch (Balandat et al., 2020).

	α	γ
MNIST	$1e-5$	Layer 1
F-MNIST	$1e-5$	Layer 1
STL10	$1e-5$	Layer 1
CIFAR10	$4e-6$	AvgPool
CIFAR100	$1e-5$	Layer 3
CoLA	$5e-6$	1
RTE	$1e-7$	11
MRPC	$1e-7$	11
SST-2	$4.132e-5$	10

6.3. Discussion

In this paper we have presented several fundamental geometric properties that occur during the TPT in intermediate model layers. We believe that more additional research can be done using these methods. We point out two additional perspectives when looking at the results given here:

(i) **First-Layer NCC mismatch may suggest at the dataset "dificulty":** When looking at NCC mismatch in early layers of the network, there is an interesting thought experiment that can be suggested. On one end, if the beginning layers have a very low NCC mismatch during the TPT - this means that the network is already achieving very good class clustering early on. The earlier this happens, the less capacity the model has

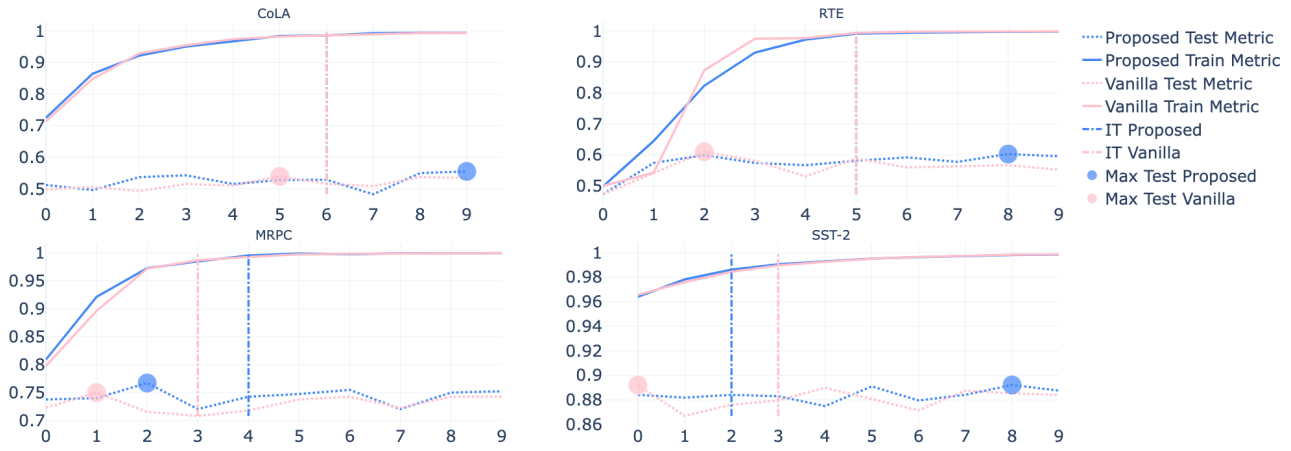

Figure 5. Optimization Procedures for Sequence-Classification experiments: **CoLA**, **RTE**, **MRPC**, **SST-2**. The SVSL achieves higher performance(test and train) at most epochs.

to achieve this clustering. When looking at Figure 2 (MNIST) - we see that in the first layer, we already have a mismatch of ~ 0.16. Since MNIST is a very simple task, this might be intuitive. However, when looking at Figure 2 (CIFAR10) - we see that the first layer only reaches 0.6 mismatch. This again, is intuitive as CIFAR10 is a hard task and we would not expect a few layers to be enough to properly cluster the features. This can be shown in several of the graphs along this paper. Perhaps this notion of thinking can aid in defining a concept of "dataset dificulty" for a certain model architecture.

(ii) **NCC-Collapse may be useful for efficient inference in large models:** In most experiments shown in the paper, the NCC-Collapse does not happen solely in the final model layer. In fact, in some architectures the collapse propagates a few layers back in the network. Suppose that for a trained network, the collapse occurs from all layers after layer j. This means that in order to get the prediction of the model on a new sample, we only need run a forward pass up to the j-th layer, and find the nearest train class-means (which needs to be computed once). In very deepnets, this can result in more efficient inference time.

We hope that these points and others shown in this paper can encourage researchers to explore further the geometrical phenomena in intermediate layers.

7. Conclusion

In this paper, we expand the notion of NCC-Mismatch as proposed in (Papyan et al., 2020). We describe how looking at intermediate layers of the network can assist in understanding the geometric phenomena that is Neural Collapse. This paper further expands these notions to NLP tasks, and shows common structure in the different modalities. We also show how encouraging inner-layer class-center consistency can assist in the training and generalization. We hope further research using these methods can continue to enrich the study in deepnets and their training paradigms.

References

Alain, G. and Bengio, Y. Understanding intermediate layers using linear classifier probes. 2017.

Allen-Zhu, Z. and Li, Y. Backward feature correction: How deep learning performs deep learning. *ArXiv*, abs/2001.04413, 2020.

Balandat, M., Karrer, B., Jiang, D. R., Daulton, S., Letham, B., Wilson, A. G., and Bakshy, E. BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization. In *Advances in Neural Information Processing Systems 33*, 2020. URL http://arxiv.org/abs/1910.06403.

Baldock, R. J. N., Maennel, H., and Neyshabur, B. Deep learning through the lens of example difficulty. *ArXiv*, abs/2106.09647, 2021.

Bartlett, P. L., Montanari, A., and Rakhlin, A. Deep learning: a statistical viewpoint. *Acta Numerica*, 30:87 – 201, 2021.

Belkin, M. Fit without fear: remarkable mathematical phenomena of deep learning through the prism of interpolation. *Acta Numerica*, 30:203 – 248, 2021.

Ben-Shaul, I. and Dekel, S. Sparsity-probe: Analysis tool for deep learning models. *ArXiv*, abs/2105.06849, 2021.

Caron, M., Touvron, H., Misra, I., J'egou, H., Mairal, J., Bojanowski, P., and Joulin, A. Emerging properties in self-supervised vision transformers. *ArXiv*, abs/2104.14294, 2021.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. E. A simple framework for contrastive learning of visual representations. *ArXiv*, abs/2002.05709, 2020.

Chollet, F. *Deep Learning with Python*. Manning Publications Co., USA, 1st edition, 2017. ISBN 1617294438.

Cohen, U., Chung, S., Lee, D. D., and Sompolinsky, H. Separability and geometry of object manifolds in deep neural networks. *Nature Communications*, 11:746, February 2020. doi: 10.1038/s41467-020-14578-5.

Dai, Z., Liu, H., Le, Q. V., and Tan, M. Coatnet: Marrying convolution and attention for all data sizes. *ArXiv*, abs/2106.04803, 2021.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *ArXiv*, abs/1810.04805, 2019.

Han, X. Y., Papyan, V., and Donoho, D. L. Neural collapse under mse loss: Proximity to and dynamics on the central path. *ArXiv*, abs/2106.02073, 2021.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

Huang, G., Liu, Z., and Weinberger, K. Q. Densely connected convolutional networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2261–2269, 2017.

Khosla, P., Teterwak, P., Wang, C., Sarna, A., Tian, Y., Isola, P., Maschinot, A., Liu, C., and Krishnan, D. Supervised contrastive learning. *ArXiv*, abs/2004.11362, 2020.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692, 2019.

Liu, Y.-H. and Arik, S. O. Explaining deep neural networks using unsupervised clustering. *ArXiv*, abs/2007.07477, 2020.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. In *ICLR*, 2019.

Marcel, S. and Rodriguez, Y. Torchvision the machine-vision package of torch. In *Proceedings of the 18th ACM International Conference on Multimedia*, MM '10, pp. 1485–1488, New York, NY, USA, 2010. Association for Computing Machinery. ISBN 9781605589336. doi: 10.1145/1873951.1874254. URL https://doi.org/10.1145/1873951.1874254.

Montavon, G., Braun, M. L., and Müller, K.-R. Kernel analysis of deep networks. *J. Mach. Learn. Res.*, 12: 2563–2581, 2011.

Nakkiran, P., Kaplun, G., Bansal, Y., Yang, T., Barak, B., and Sutskever, I. Deep double descent: where bigger models and more data hurt. *Journal of Statistical Mechanics: Theory and Experiment*, 2021, 2020.

Papyan, V., Han, X. Y., and Donoho, D. L. Prevalence of neural collapse during the terminal phase of deep learning training. *Proceedings of the National Academy of Sciences of the United States of America*, 117:24652 – 24663, 2020.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019.

Power, A., Burda, Y., Edwards, H., Babuschkin, I., and Misra, V. Grokking: Generalization beyond overfitting on small algorithmic datasets. *ArXiv*, abs/2201.02177, 2022.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., and Sutskever, I. Learning transferable visual models from natural language supervision. In *ICML*, 2021.

Raghu, M., Unterthiner, T., Kornblith, S., Zhang, C., and Dosovitskiy, A. Do vision transformers see like convolutional neural networks? *ArXiv*, abs/2108.08810, 2021.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2015.

Tian, K., Zhou, S., and Guan, J. Deepcluster: A general clustering framework based on deep learning. In *ECML/PKDD*, 2017.

Vaswani, A., Shazeer, N. M., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. *ArXiv*, abs/1706.03762, 2017.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. R. Glue: A multi-task benchmark and analysis platform for natural language understanding. *ArXiv*, abs/1804.07461, 2018.

Weng, L. Contrastive representation learning. *lilianweng.github.io/lil-log*, 2021. URL https://lilianweng.github.io/lil-log/2021/05/31/contrastive-representation-learning.html.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T. L., Gugger, S., Drame, M., Lhoest, Q., and Rush, A. M. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45, Online, October 2020. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2020.emnlp-demos.6.

Xie, J., Girshick, R. B., and Farhadi, A. Unsupervised deep embedding for clustering analysis. *ArXiv*, abs/1511.06335, 2016.

Zbontar, J., Jing, L., Misra, I., LeCun, Y., and Deny, S. Barlow twins: Self-supervised learning via redundancy reduction. In *ICML*, 2021.

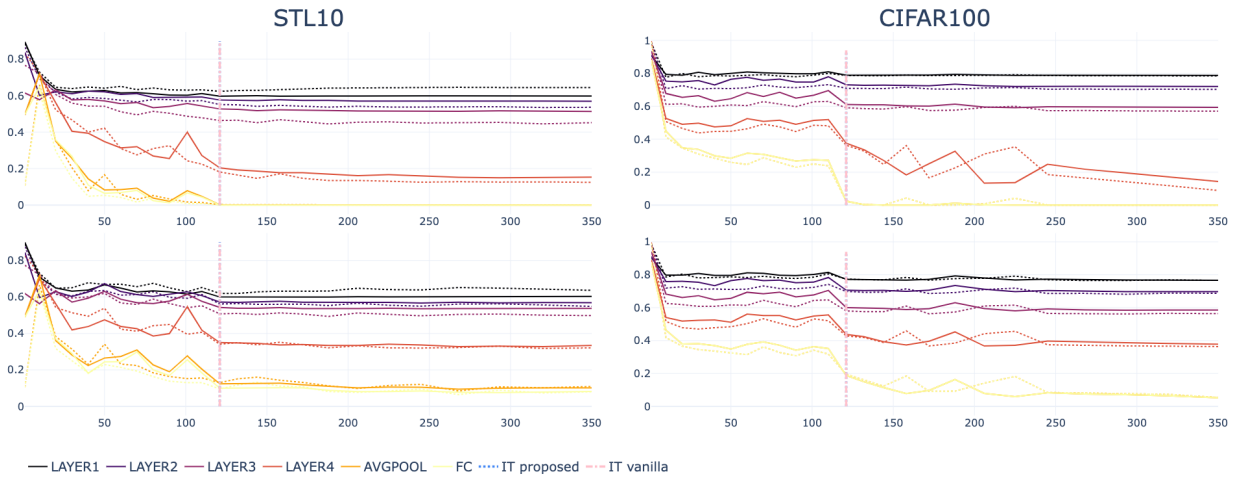

Figure 6. NCC mismatch for Vision Classification Datasets: **STL10** and **CIFAR100**, using both vanilla (**Solid**) and SVSL (**dashed**) losses. **Top:** train NCC mismatch, **Bottom:** test NCC mismatch. The IT for each of the losses is shown using a vertical dash-dot line.

A. Additional Results

In Figure 6 we show the NCC mismatch for the STL10 and the CIFAR100 Vision datasets. We see that these plots also show the characteristics shown in other datasets. We also see that for these datasets, the NCC mismatch in early layers of the network are higher. This matches the Discussion-Point 6.3, as these datasets are considered more difficult than MNIST, Fashion-MNIST, and CIFAR10.

Lastly, we show an example implementation of the SVSL in Code-Listing 1. In this example, we make use of the hidden states given in Transformers. Alternatively, the supplementary code makes use of PytorchHooks for the same purpose.

Code Listing 1. Example implementation of the SVSL in PyTorch code

```
def SVSL(outputs, targets, C, gamma, alpha):
    '''
    Computes the SVSL loss for batch.
    Parameters:
    outputs : model outputs for batch
    targets : ground truth labels for batch
    C : number of classes
    gamma : hyperparameter for first layer used
    alpha : SVSL weighing factor

    Returns: computed loss
    '''
  svsl = 0.
  for layer_idx in range(gamma, len(outputs.hidden_states) - 1):
    intermediate_features = outputs.hidden_states[layer_idx]
    for cur_class in range(1, C):
      indices = torch.where(targets == cur_class)[0]
      cur_loss = 0
      if len(indices) == 0:
        continue
      class_mean = torch.mean(intermediate_features[indices], dim=0)
      for j in range(len(indices)):
        cur_loss += torch.sum(
        (intermediate_features[indices[j]] - class_mean) ** 2)
      cur_loss /= len(indices)
      svsl += cur_loss
```

```
svsl /= C
return alpha * svsl
```