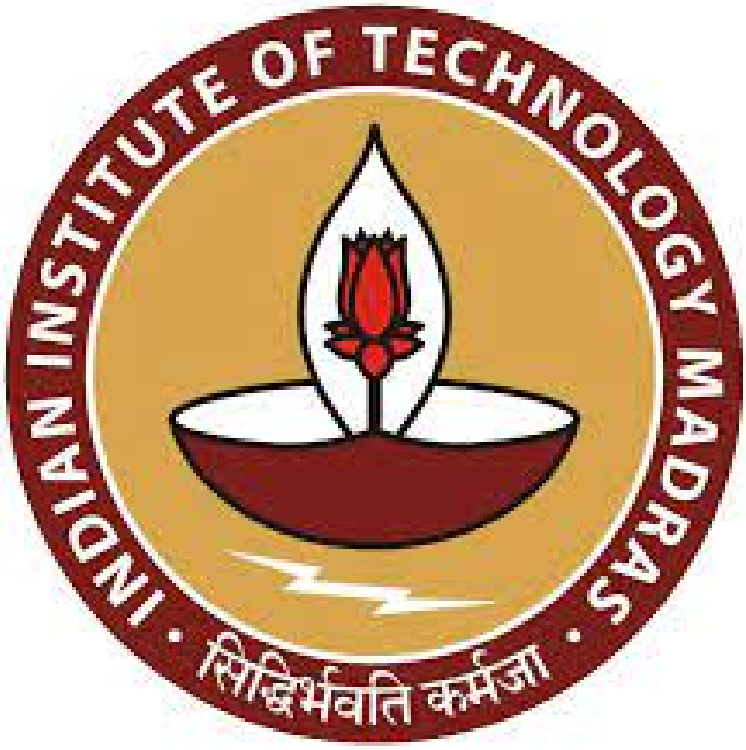
EE5179 Project : Neural collapse in hierarchical loss functions

Nikhil Anand (BE20B022) Ruban Vishnu Pandian (EE19B138)
Siddharth Betala (BE19B032)

January 18, 2024



Contents

1 Introduction

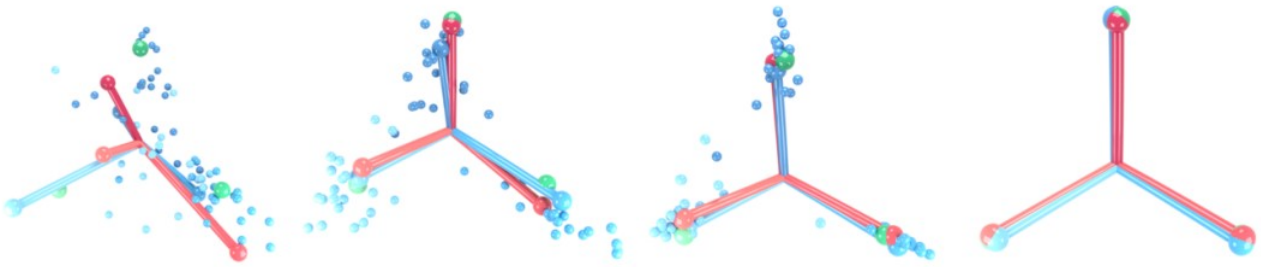
Deep neural networks of the current era have achieved significant performance on tasks like image classification, object detection, etc. Much of the analysis done in this field is empirical. One such empirical phenomenon is **Neural collapse** (NC). When a neural network is trained beyond convergence, it reaches a state known as **Terminal phase of training** (TPT). In this state, essentially the final hidden layer values concentrate around the class means and hence, the final layer simply follows the nearest class-mean mapping rule.

Neural collapse emerges in various deep models as discussed in [1]. But an interesting experiment to try out is to check whether they appear in models which use **Hierarchical loss functions** (HLF). In this academic report, the background theory of NC, their emergence in some NN models and experiments with HLF based models have been discussed in detail.

2 Mathematical theory behind neural collapse

Neural collapse is an empirical phenomenon which occurs in deep classification models when they are trained beyond zero training loss which is also known as **Terminal phase of training (TPT)**. It is described as collective emergence of four sub-phenomena which are described below:

- **Collapse of variability:** Data samples essentially collapse to the class mean and intra-class variability collapses to zero.
- **Formation of simplex equiangular tight frame (ETF):** The class means tend to form an ETF, i.e., all lie on an m-dimensional hypersphere, are linearly separable and are at maximally equal angles from each other.
- **Self-dual alignment:** The final layer class weight vectors also form an ETF which converges to the class means ETF upto rescaling.
- **Nearest class-mean decoding:** To classify a test point, simply find the nearest class-mean to its final layer activation vector and declare that class as output.



Emergence of NC with training [1]

To quantify these phenomena, certain metrics are developed. As training progresses, these metrics tend to zero signifying the emergence of NC. Those metrics are explained below:

- **Collapse of variability metric:** Assuming a balanced dataset of K classes with n training examples in each class, let us call the penultimate layer feature vector for the i^{th} example of k^{th} class as $f_{k,i}$ and the class mean as $\mu_k = \frac{1}{n}\sum_{i=0}^{n} f_{k,i}$. Also, let us call the global mean as $\mu_g = \frac{1}{Kn}\sum_{k=1}^{K}\sum_{i=1}^{n} f_{k,i}$. We consider the following two matrices:

  1. **Intra-class covariance:** $\Sigma_w = \frac{1}{Kn}\sum_{k=1}^{K}\sum_{i=1}^{n}((f_{k,i}-\mu_k)(f_{k,i}-\mu_k)^T)$
  2. **Inter-class covariance:** $\Sigma_b = \frac{1}{K}\sum_{k=1}^{K}((\mu_k-\mu_g)(\mu_k-\mu_g)^T)$

  We define the NC1 metric as:

  $$NC1 := \frac{1}{K} tr(\Sigma_w \Sigma_b^{\dagger})$$

  where $A^{\dagger}$ and $tr(A)$ denote the pseudo-inverse and trace of matrix A respectively. This metric captures how invariant the intra-class feature vectors are and how vastly variant the inter-class means are.

- **Formation of simplex ETF metric:** Let the matrix formed by the global-mean centered class mean vectors as rows be $M \in \mathbb{R}^{K\times m}$. A simplex ETF is said to be formed if the centered mean vectors follow this property:

  $$cos(\mu_k-\mu_g, \mu_{k'}-\mu_g) = \frac{<\mu_k-\mu_g, \mu_{k'}-\mu_g>}{||\mu_k-\mu_g||_2||\mu_{k'}-\mu_g||_2} = -\frac{1}{K-1} \text{ for } k \neq k'$$

  i.e., we ideally should have the following equality:

  $$MM^T = \frac{K}{K-1}I_K - \frac{1}{K-1}1_K 1_K^T$$

  where 1_K is a K-column, all-one vector. Based on this equality, the NC2 metric is designed as follows:

  $$NC2 := \left|\left| \frac{MM^T}{||MM^T||_F} - \frac{1}{\sqrt{K-1}}(I_K - \frac{1}{K}1_K 1_K^T) \right|\right|_F$$

  where $||A||_F$ denotes the Frobenius norm of matrix A.

- **Self-dual alignment metric:**Let the last layer weights matrix be $A \in \mathbb{R}^{K\times m}$. This phenomenom states:

  $$\frac{A}{||A||_F} = \frac{M}{||M||_F}$$

  i.e., the weights align with the class means upto a rescaling factor. Hence, the NC3 metric is defined as:

  $$NC3 := \left|\left| \frac{AM^T}{||AM^T||_F} - \frac{1}{\sqrt{K-1}}(I_K - \frac{1}{K}1_K 1_K^T) \right|\right|_F$$

- **Nearest class-mean decoding metric:** Let's say the class predicted by the network for the i^{th} example be c_i. If NC has occurred then this predicted class should be the same as the class whose class mean vector is the nearest to the penultimate feature vector f_i. Hence, the NC4 metric is designed as:

  $$NC4 := \frac{1}{Kn}\sum_{i=1}^{Kn}\mathbb{I}(c_i \neq \underset{k}{\text{argmin}}\, ||f_i-\mu_k||_2)$$

  where $\mathbb{I}(X) \in 0, 1$ is the indicator operator which takes the value 1 when the statement inside it is true and vice-versa.

3 Concepts behind Hierarchical Neural Networks

The goal of a hierarchical neural network is to not only minimise the number of mistakes and therefore improve accuracy of classification, but to also minimise the severity of mistakes. We've added some necessary definitions below:

1. **Top-k prediction:** The predictions with the k highest probabilities.

2. **Hierarchical distance of a mistake:** Mean height of the Least Common Ancestor (LCA) between the ground truth and the predicted class in the class hierarchy. It's a measure of the severity of a mistake.

3. **Hierarchical Average Top-k Error:** The hierarchical average top-k error takes the mean LCA height between the ground truth and each of the k most likely classes.

Class hierarchies need to be incorporated into the training process in some way to ensure that these metrics of severity of mistakes are minimised. We incorporate them into the loss function in some way.

$$\frac{1}{N}\sum_{i=1}^{N}\mathcal{L}(\phi(x_i;\theta),y(C_i))+\mathcal{R}(\theta)$$

1. **Label embeddings** y^H: Embeddings could be made in such a way that cosine similarity of embeddings is proportional to the hierarchical distance between the classes. The new loss function would be:

$$\frac{1}{N}\sum_{i=1}^{N}\mathcal{L}(\phi(x_i;\theta),y^H(C_i))+\mathcal{R}(\theta)$$

2. **Hierarchical loss function** $\mathcal{L}^H$: The loss function itself could be modified, for example, by adding a penalty based on the severity of the mistake. We could also weigh probabilities in the Cross-Entropy loss function according to class similarity. The new loss function would be:

$$\frac{1}{N}\sum_{i=1}^{N}\mathcal{L}^H(\phi(x_i;\theta),y(C_i))+\mathcal{R}(\theta)$$

3. **Hierarchical architecture** ϕ^H: By altering the hierarchical architecture, we can incorporate more generic class distinctions in earlier layers and provide more fine-grained distinctions in later layers. The new loss function would be:

$$\frac{1}{N}\sum_{i=1}^{N}\mathcal{L}(\phi^H(x_i;\theta),y(C_i))+\mathcal{R}(\theta)$$

3.1 Hierarchical Cross Entropy

Hierarchical Cross Entropy is a hierarchical loss function that can be used for incorporating class hierarchies. In order to frame it, we first compute the conditional probabilities from the absolute probabilities.

$$p(C^{(l)}|C^{(l+1)})=\frac{\Sigma_{A\in Leaves(C^{(l)})}p(A)}{\Sigma_{B\in Leaves(C^{(l+1)})}p(B)}$$

The loss function is:

$$\mathcal{L}_{HXE}(p,C)=-\sum_{l=0}^{h-1}\lambda(C^{(l)})\log p(C^{(l)}|C^{(l+1)})$$

Thus, we move upwards along the leaf node and weight conditional probabilities to compute the loss function.

$$\lambda(C) = \exp(-\alpha h(C))$$

Weights are computed based on the height of the parent node. Thus, the higher the parent, the lower the weight it holds in the calculation of loss.

3.2 Soft Labels

Instead of using conditional probabilities of all classes and parent classes, we can instead only consider the leaf node and provide some "soft" labels for the cross entropy function rather than simple one-hot encodings.

$$\mathcal{L}_{soft} = -\sum_{A \in C} y_A^{soft}(C) \log p(A)$$

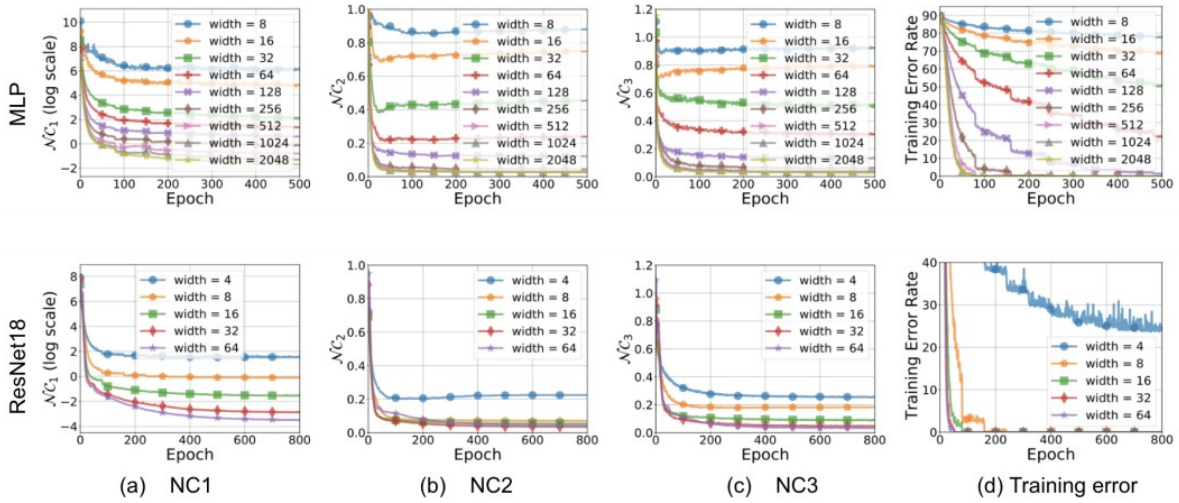
The soft labels allow us to consider not only the probability of the true class but the probabilities of all the classes weighted according to their distance from the true class. We define these soft labels (weights) as follows:

$$y_A^{soft}(C) = \frac{\exp(-\beta d(A, C))}{\Sigma_{B \in C} \exp(-\beta d(B, C))}$$

$$d(A, C) = \frac{height(LCA)}{height(tree)}$$

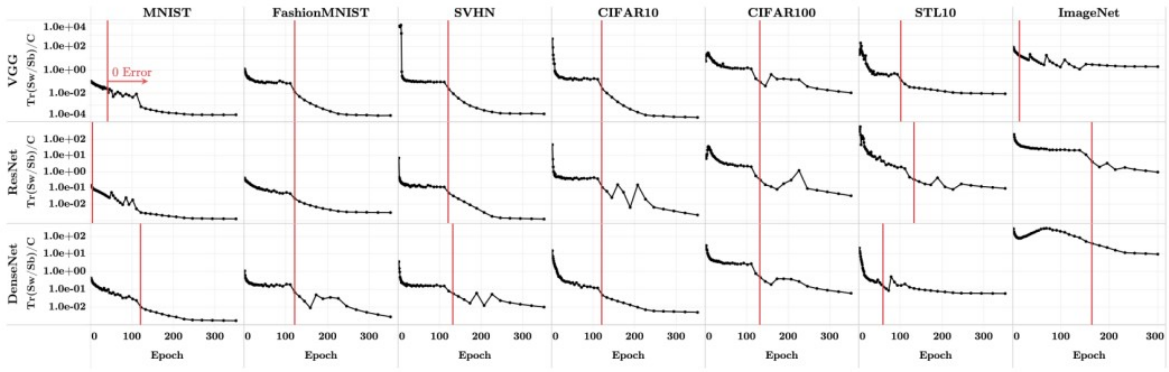
4 Plots and results:

In the paper, results have been shared for multiple model-dataset combinations. Those plots are shared below:



NC in MLP and ResNet18 with CIFAR-10 [1]

We replicated the same experiments done in [1] to understand the occurrence of NC with two main loss functions: **MSE loss** and **Cross entropy loss**. Those results are shared below:



Occurrence of NC in multiple model-dataset combinations [1]

5 References

[1] Kothapalli, V., Rasromani, E., & Awatramani, V. (2022). Neural Collapse: A Review on Modelling Principles and Generalization. *Trans. Mach. Learn. Res., 2023*

[2] Bertinetto, L., Mueller, R., Tertikas, K., Samangooei, S., & Lord, N.A. (2019). Making Better Mistakes: Leveraging Class Hierarchies With Deep Networks. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 12503-12512.*