Geometric interpretation of a CNN's last layer

Alejandro de la Calle, Aitor Aller, Javier Tovar and Emilio J. Almazán
Image Recognition Team, Nielsen
c/ Salvador de Madariaga 1, 28027, Madrid
corresponding author: alejandro.delacalle.consultant@nielsen.com

Abstract

Training Convolutional Neural Networks (CNNs) remains a non-trivial task that in many cases relies on the skills and experience of the person conducting the training. Choosing hyper-parameters, knowing when the training should be interrupted, or even when to stop trying training strategies are some difficult decisions that have to be made. These decisions are difficult partly because we still know little about the internal behaviour of CNNs, especially during training. In this work we conduct a methodical experimentation on MNIST public database of handwritten digits to better understand the evolution of the last layer from a geometric perspective: namely the classification vectors and the image embedding vectors. Within this context we present the problem of the variability across equal set-up trainings due to the random component of the initialisation method. We propose a novel approach that guides the initialisation of the parameters in the classification layer. This method reduces 12% the variability across repetitions and leads to accuracies 18% higher on average.

1. Introduction

To this day nobody doubts about the potential of Deep Learning for addressing Artificial Intelligence (AI) challenges. Moreover, the Computer Vision field has experienced a revolution where Deep Learning models have substantially outperformed the state of the art, not only in image classification and detection but also in other domains such as image processing, 3D modelling or Natural Language Processing. Despite its success, an important fraction of the community has strongly criticised the inability to provide a clear explanation of how CNNs work inside. Important research has been conducted for the visualisation of the filters and the activations [12, 8, 2]. These works provide tools that enable better diagnosis for addressing issues and identifying failure modes. However, we are still far from a good understanding of neural nets, especially during training time.

Apart from high level metrics such as accuracy or cross entropy loss, we do know little about how the dynamics of filters and classifiers of the model evolve during training. For instance, it would be helpful to know information about the distribution of the image features in the latent space, or even if the parameters have been initialised to locations that can ensure good convergence. By better understanding the behaviour of the model insights and how it evolves during training we should expect better training strategies that improve the accuracy.

Latent features and loss strategies. In order to extend the understanding of CNNs is common to divide the CNN into two blocks: namely the block in charge of extracting features of interest from the input image, also known as embedding or encoding, and the classifier block that receives the embedding and predicts the correct class. A well trained network is expected to generate similar embeddings for images that belong to the same class and dissimilar embeddings for different classes. Generating good representative ans discriminative features is chief to ensure good accuracies on image classification and image retrieval tasks. Training a CNN for image classification with the standard cross entropy loss does not ensure good separability of classes in the embedding space [5, 6]. A common approach is to train directly the embedding space using pairs of images [9] or triplets [7]. These losses tend to obtain more discriminative embeddings than the standard cross entropy loss. However, since the number of possible pairs/triplets explode with the size of the dataset, these methods require a non-trivial process of data mining to generate the pairs or triplets of interest for training. A popular approach to avoid data mining is the center loss [11] yet it requires extra computation to re-calculate the class centres and intra-class distances at every iteration. Alternatively, some researchers propose variations of the cross entropy loss that aim at reducing the intra-class distance and increasing the inter-class distance. The work of W. Liu et al. [5] adds a margin in the angle of the embedding vector with respect to the correct classifier, in a similar way that Hinge loss enforces maximum margins between embeddings and classifier's bound-

aries. Furthermore, R. Ranjan et al. [6] noted that the ℓ^2-norm of the feature vectors is a good indicator of the representativeness of the image to its class. They proposed the Crystal loss, which computes the cross entropy loss over features where all have the same norm. Another interesting approach was proposed by W. Wan et al. [10] where the embedding space is modelled as a mixture of Gaussian distributions. The loss function aims at increasing the probability of each instance to its distribution.

Parameters initialisation. A significant amount of work has been conducted on the initialisation of parameters and how they can help on mitigating the exploding or vanishing gradients, as well as to avoid slow convergence [3, 4]. In the work of B. Ayinde et al. [1] a study was conducted on how the initialisation methods affected the amount of redundant filters learned. These well known techniques have a random component that makes each training start from a different configuration and likely to lead to different training states. Thus, it is worth studying deeper the variability across training repetitions and how this variability can be reduced.

Using the previous research as the seed of our study we investigated how the backbone of the network and the final classification layer evolve during training. From a geometric point of view, we treat the classifier's weights as vectors that live in the embedded space. This perspective allows us to focus on the geometric evolution of both vectors representing the classifier weights and image embeddings. We conduct a series of ablation studies to better understand the interplay between these vectors. Moreover, we explore the variability across initialisations in unbalanced datasets with a long tail shape. Finally, we propose a novel initialisation of the classifiers vectors based on the train set distribution. Hence, this method reduces the variability in 12% across initialisations in a long-tail version of MNIST.

This paper is arranged as follows: Section 2 introduces some background and the geometric approach of this study. In Section 3 we identify issues associated with standard training techniques and present a method that mitigates these issues through a guided initialization of the last layer vectors. Lastly, Section 4 presents some conclusions and further work.

2. Background

Convolutional Neural Networks (CNNs) can be divided in two blocks, as shown in Figure 1, namely the **feature extraction block** and the **classification block**:

- **The feature extraction block** receives an input image and applies a series of convolutions and pooling operations with the goal of identifying discriminative features. The output of this block is a one-dimensional vector regarded as the image encoding or embedding of the image. If it is well trained we should expect encodings from the same class to be close together. These vectors are the input for the classification block.

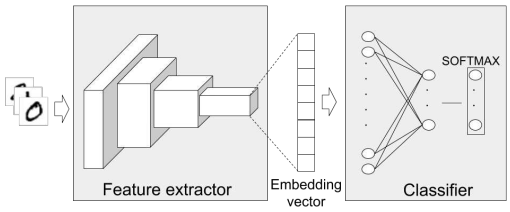


Figure 1: Schematic diagram of a convolutional neural network. The network is mainly composed by two parts: the backbone block and the classifier block.

- **The classification block** is a classifier with the softmax function. Although the classification block can in general be composed of several dense layers, we will refer to classification block as the last layer of the CNN throughout the paper. This layer calculates the class probability for the input image. It has a linear classifier that performs the linear transformation given by

$$z_c = \sum_{j=1}^{N} W_{c,j} \cdot x_j + b_c, \tag{1}$$

where $\boldsymbol{W} \in \mathbb{R}^{C \times N}$ is the classifier weight matrix with C being the number of classes and N the size of the image encoding, $\boldsymbol{b} \in \mathbb{R}^C$ is the bias term, $\boldsymbol{x} \in \mathbb{R}^N$ is the image embedding i.e. the outcome of the feature extraction block, and $\boldsymbol{z} \in \mathbb{R}^C$ is the prediction class vector. We remove the bias term $\boldsymbol{b}$ throughout for simplicity. The block also uses the softmax function, which is a non-linear transformation that produces a probability distribution across all classes. This function $f(\boldsymbol{x})$ is defined as

$$[f(\boldsymbol{z})]_c = \frac{\exp(z_c)}{\sum_{c=1}^{C} \exp(z_c)}, \tag{2}$$

where $[f(\boldsymbol{z})]_c$ is the probability of the c^{th} class. Note that the performance of the classifier block is highly dependent upon the quality of the features. The classifier will benefit from a well separated class-wise features.

During training the network tries to optimise a loss function through back-propagation and gradient descent. One of the most common objective functions is the cross-entropy loss, which measures the difference between the predicted distribution $f(\boldsymbol{x})$ and the target distribution $p(\boldsymbol{x})$ (i.e. the one constructed from the ground truth). For a given instance

the cross-entropy is expressed as follows:

$$\mathcal{L}_i = -\sum_{c=1}^{C} \left[p(\boldsymbol{x}^{(i)})\right]_c \log \left[f(\boldsymbol{x}^{(i)})\right]_c, \tag{3}$$

where C is the total number of classes.

2.1. Geometric interpretation

We can express Eq. (3) using the geometric notation of the dot product as follows

$$\mathcal{L}_i = -\sum_{c=1}^{C} \left[p(\boldsymbol{x}^{(i)})\right]_c \times \\ \times \log \left[\frac{\exp\left(\|\boldsymbol{W}_c\| \cdot \|\boldsymbol{x}^{(i)}\| \cos\theta_c^{(i)}\right)}{\sum_{c=1}^{C} \exp\left(\|\boldsymbol{W}_c\| \cdot \|\boldsymbol{x}^{(i)}\| \cos\theta_c^{(i)}\right)}\right], \tag{4}$$

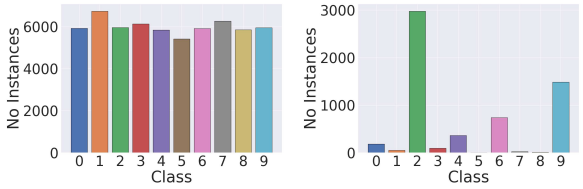
where $\theta_c^{(i)}$ is the angle between the image encoding (i) with respect to the classifier vector c. Considering Eq. (4), we can see that there are two pathways to reduce the loss according to the two blocks of the network: by updating the parameters of the backbone, i.e. the feature extraction block, or by updating the parameters of the classifier.

- **Backbone update:** This entails updating parameters of the convolutional filters of the network, eventually leading to different encoding vectors. From the geometric perspective, in order to reduce the loss the network can reduce the angle $\theta_c^{(i)}$ by moving the encoding closer to its classifier.

- **Classifier update:** Updating the classifier block entails updating the classifier's vectors. The training process can yield an increase of $|W_c|$ for the correct class or/and change the direction of this vector, so the angle $\theta_c^{(i)}$ of the correct class is reduced. Likewise, it can also reduce the norm of the rest of the classifiers or/and increase their angles with the encoding by changing their directions away from it.

3. Experiments and results

In the following experiments we study the interplay between the image encodings and the classifier layer during training. In particular we explore these dynamics in a dataset with an unbalanced distribution of instances per class. In our experiments we use as backbone the ResNet 101 architecture [4] with an embedding of length 2, a batch size of 512, an initial learning rate of 5×10^{-4} that gets divided by 5 at epochs 15^{th} and 80^{th}. We use a weight decay of 5×10^{-3}, the ADAM optimiser and Xavier for the initialisation of parameters.



(a) Balanced dataset (b) Unbalanced dataset

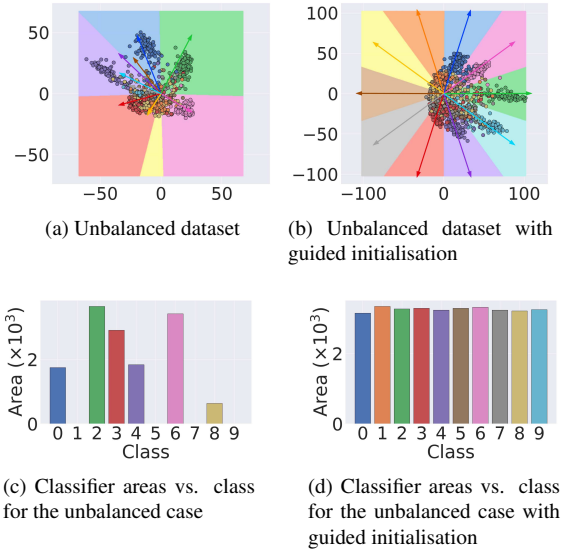
Figure 2: Number of instances per class in the dataset, in case of (a) a balanced dataset, and (b) and unbalanced dataset.

3.1. Unbalanced dataset

Unbalanced datasets are of great interest due to its presence in real-world problems. A particular case of unbalancing is the long-tail dataset. We have modified the MNIST dataset in a way that the instances are geometrically distributed across classes following the relation $y_c^{LT} = y_c * g_c$ where y_c is the number of instances of class c in the balanced dataset, and the down-sampling factor g_c is given by

$$g_c = p(1-p)^c \quad c = 1, \ldots, C. \tag{5}$$

In this study we have set $p = \frac{1}{2}$. The resulting distribution is shown in Figure 2b. Note that only the train set has been unbalanced, the test set remains balanced as in the original distribution of MNIST (Figure 2a).



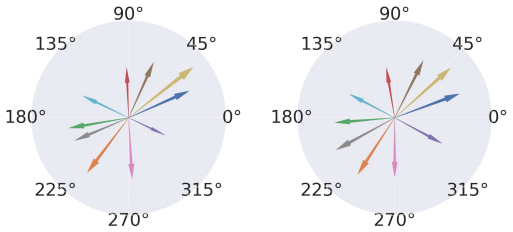
(a) Unbalanced dataset (b) Unbalanced dataset with guided initialisation

(c) Classifier areas vs. class for the unbalanced case (d) Classifier areas vs. class for the unbalanced case with guided initialisation

Figure 3: (a) and (b) Embedded space and classifier areas for each class respectively, at the end of a training without restrictions for the unbalanced dataset case. (c) and (d) are the same as (a) and (b) but using guided initialisation.

Figures 3a and 3c show the resulting embedded space

from a training with our unbalanced dataset. The single most striking observation is that some classes (i.e. 1, 5, 7 and 9) finish the training without classification area. Interestingly, these classes correspond to classes with low representation in the train set. We also observe that classes with higher presence in train tend to overcome the adjacent classes with less presence, up to a point where the minority classes are left without area.
Furthermore, we have seen in our experiments that the direction of the classification vectors withstand little variation during the course of training, as depicted in Figure 4.



(a) Initial state of the classifiers' vectors

(b) Final state of the classifiers' vectors.

Figure 4: State of the classifiers' vectors during training time.

The previous observations evidence the importance of the initialisation, especially in long-tail datasets. The evolution of the accuracy for three different experiments that have been randomly initialised reveals high variability among repetitions, where each training leads to different accuracies, from 95% to 64% in train and from 50% to 33% in test. A difference of 31% and 17% in train and test respectively for trainings with the same set of hyperparameters and number of epochs.

To mitigate this effect, we propose a novel approach that consists on a guided initialisation where classification vectors of classes with similar number of instances are located next to each other. This approach reduces the competition between high and low-represented classes, and therefore the areas of the former will not push out of the embedded space the areas of the latter.

In Figures 3b and 3d the results of the guided initialisation are shown. We observe how the absent areas of the previous experiment are now present in the embedded space. In addition, variability among trainings has been reduced to 4% and the final test accuracies are 18% higher in average. Hence, trainings using guided initialisation turns out to be more robust, especially when training unbalanced datasets.

4. Conclusions and Future Work

In this work we have studied from a geometric perspective the dynamics of CNNs during training time. Specifically, we have explored the interplay between classification and image encoding vectors in the final layer space as the training progresses.

Furthermore, we have shown how unbalanced datasets are highly sensitive to the randomness of the initialisation, reporting up to 17% accuracy difference in test across repetitions. We propose a novel approach to initialise the classification layer parameters that reduces this variability to 4%. This method sets the initial direction of the vectors in a way that the competition for the classification area happens between classes with similar number of training instances. Hence, minimising the risk of absent areas for classes with less presence in train. Moreover, this method yielded accuracies 18% higher in average, suggesting that it sets more robust initial states that lead with more frequency to good local minima.

References

[1] B. O. Ayinde, T. Inanc, and J. M. Zurada. On Correlation of Features Extracted by Deep Neural Networks. *arXiv e-prints*, page arXiv:1901.10900, Jan. 2019. 2

[2] A. Binder, S. Bach, G. Montavon, K.-R. Müller, and W. Samek. Layer-wise relevance propagation for deep neural network architectures. In *ICISA*. Springer, 2016. 1

[3] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. Proceedings of Machine Learning Research. PMLR, 2010. 2

[4] K. He, X. Zhang, S. Ren, and J. Sun. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. *arXiv e-prints*, 2015. 2, 3

[5] W. Liu, Y. Wen, Z. Yu, and M. Yang. Large-Margin Softmax Loss for Convolutional Neural Networks. *arXiv e-prints*, page arXiv:1612.02295, Dec. 2016. 1

[6] R. Ranjan, A. Bansal, H. Xu, S. Sankaranarayanan, J.-C. Chen, C. D. Castillo, and R. Chellappa. Crystal Loss and Quality Pooling for Unconstrained Face Verification and Recognition. *arXiv e-prints*, 2018. 1, 2

[7] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the CVPR*, pages 815–823, 2015. 1

[8] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *IEEE ICCV*, 2017. 1

[9] Y. Sun, Y. Chen, X. Wang, and X. Tang. Deep learning face representation by joint identification-verification. In *Advances in neural information processing systems*, 2014. 1

[10] W. Wan, Y. Zhong, T. Li, and J. Chen. Rethinking Feature Distribution for Loss Functions in Image Classification. *arXiv e-prints*, page arXiv:1803.02988, Mar. 2018. 2

[11] Y. Wen, K. Zhang, Z. Li, and Y. Qiao. A discriminative feature learning approach for deep face recognition. In *ECCV*, pages 499–515. Springer, 2016. 1

[12] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV*, pages 818–833. Springer, 2014. 1