Few-Shot Learning with Online Triplet Mining

Prithvi Raju Nagalapelli
Computer Science and Information Management
Asian Institute of Technology
Bangkok, Thailand
nprithviraj24@gmail.com [1] st120000@ait.asia [2]

***Abstract*—The process of classifying images in deep learning applications can be computationally expensive and extremely challenging due to numerous constraints such as requirement of huge data in order to train the model. A prototypical example of this is the *few-shot learning* setting, in which we must correctly make predictions given only few examples of each new class. In this paper, I propose an encoder that maps an image to an embedding space, followed by metric learning to classify distinct images based on its location in the space. The proposed methodology uses triplet loss from *FaceNet [1]* which can learn to aggregate two examples with the same label have their embeddings close together in the embedding space, and divide two examples with different labels have their embeddings far from each other. This paper proposes an optimal way called as Online Triplet Mining that selects a right triplet for each embedding which can save extra computations.**

***Index Terms*—few-shot learning, metric learning, encoder, embedding space, triplet loss**

I. Introduction

Typically, classification involves fitting a model given many examples of each class, then using the fit model to make predictions on many examples of each class. Few shot learning is a novel-method for classification where the system has the ability to learn from few-instances of each class. This is a relatively easy problem for humans. For example, a person may see a Porsche sports car one time, and in the future, be able to recognize Porsche in new situations, on the road, in movies, in books, and with different lighting and colors.

Few-shot learning has been showcased as prominent interest in research field. Some of the methods to achieve few shot learning are metric learning, meta-learning, semantics-based methods etc. The proposed methodology to achieve few-shot learning incorporate metric learning where the objective is to learn a mapping from images to embedding space. The intuition is that images from the same category are closed together and images from different categories are far apart. The conjecture is that it holds true for an unseen image.

A network that has been popularized given its use for one-shot learning (shares the same class of problem as few-shot learning) is the Siamese network. A Siamese network is an architecture with two parallel neural networks, each taking a different input, and whose outputs are combined to provide some prediction.

II. Related Work

Metric learning methods have the advantage that they rapidly learn novel concepts without retraining.

Fig. 1. Siamese networks are the most widely used method to solve Humpback whale identification challenge on Kaggle. The data consists around 10,000 fluke images of different whales with 5000 different classes.

A. One-shot Image Recognition using Siamese Neural Networks

One-shot learning are classification tasks where many predictions are required given one (or a few) examples of each class, and face recognition is an example of one-shot learning. Siamese networks are an approach to addressing one-shot learning in which a learned feature vector for the known and candidate example are compared.

The aim of Siamese Neural Network [2] is to first learn a neural network that can discriminate between the class-identity of image pairs, which is the standard verification task for image recognition. The method hypothesize that networks which do well at a verification should generalize to one-shot classification. The verification model learns to identify input pairs according to the probability that they belong to the same class or different classes. This model can then be used to evaluate new images, exactly one per novel class, in a pairwise manner against the test image. The pairing with the highest score according to the verification model network is then awarded the highest probability for the one-shot task.

A Siamese network is used and a weighted L1 distance function is learned between their embeddings. This is done by applying L1 distance on the output embeddings then adding one fully connected layer to learn the weighted distance. The loss function used in the paper is a regularized cross entropy, where the main aim is to drive similar samples to predict 1, and 0 otherwise.

The main drawback of this method is that it uses regularised cross-entropy as loss function that compares each image with everything other image, consequently it can get computationally expensive for dataset with many classes. It's

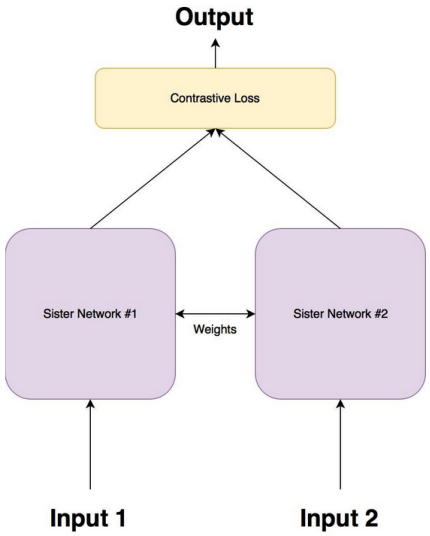

Fig. 2. There are two sister networks, which are identical neural networks, with the exact same weights. Each image in the image pair is fed to one of these networks.The networks are optimised using a contrastive loss function

optimization includes standard gradient that is passed across the twin networks due to the tied weights.

B. Contrastive Loss

One approach is to learn a mapping from inputs to vectors in an embedding space where the inputs of the same class are closer than those of different classes. Once the mapping is learned, at test time a nearest neighbors method can be used for classification for new classes that are unseen. A Siamese network is trained with the output features fed to a Contrastive Loss:

$$\mathcal{L} = (1 - Y)\frac{1}{2}D^2 + Y\frac{1}{2}max(0, m - D)$$

Y label is 0 for similar class samples, 1 for dissimilar, and D is the euclidean distance. So the loss will decrease the distance D when the samples are from the same class, on the other hand when they are dissimilar it will try to increase D with a certain margin m. The margin purpose is to neglect samples that have larger distance than m, since we only want to focus on dissimilar samples that appear to be close. This method is a predecessor of Triplet loss, that will be discussed in the next section. Figure 2 illustrates the typical Siamese Network architecture.

III. Methodology

The proposed methodology explains the pipeline to achieve few-shot learning using Encoding Networks via Triplet Loss.

A. Data Prepocessing

Some of the standard practice in data preprocessing is to remove duplicate images. Some images are perfect binary copies, while other have been altered somewhat: contrast and brightness, size, masking the legend, etc.

Two images are considered duplicate if they meet the following criteria:

1) Both images have the same Perceptual Hash (phash); or
2) Both images have:
   a) phash that differ by at most 6 bits, and;
   b) have the same size, and;
   c) the pixelwise mean square error between the normalized images is below a given threshold.

The p2h dictionary associate a unique image id (phash) for each picture. The h2p dictionary associate each unique image id to the prefered image to be used for this hash. The preferred image is the one with the highest resolution, or any one if they have the same resolution.

Irrelevant and noisy images usually don't contribute to the accuracy of the model, such images are removed by handpicking.

B. The process of Metric Learning via Encoding

Images are distinguished based on the intricate features that define the underlying objects in an image. One way to dismantle these distinguishing features and convert them is by using a convolutional neural network which consists of multiple convolution layers, where each layer compose the representations of the previous layer to learn a higher level abstraction. CNN maps from the image space into feature (embedding) space: $\mathbb{R}^d$
$Pixels \longrightarrow Edges \longrightarrow Contours \longrightarrow Objectparts \longrightarrow EmbeddingSpace.$

In order to learn the mapping of image, the proposed methodology projects an image into d-dimensional Euclidean space where distances directly correspond to a measure of similarity.

One of the most widely regarded pretrained CNN models are the ResNet architectures. The constituent of ResNet architectures are called residual blocks, that allow for parametrization relative to the identity function f(x)=x, due to skip connection. Adding residual blocks increases the function complexity in a well-defined manner, such as adding multiple residual blocks to ResNet18 resulted in ResNet34. We can train an effective deep neural network by having residual blocks pass through cross-layer data channels. ResNet had a major influence on the design of subsequent deep neural networks, both for convolutional and sequential nature. So naturally, ResNet-18/34/50 along with a fully connected layer is an optimal choice for an encoder. The fully connected layer maps to the number of dimensions in euclidean space where the value at each neuron in the last layer quantify the respective feature of the euclidean space.

C. Triplet Loss

After calculating the embeddings of each image, encoder is scrutinized based on its performance. The encoder is expected to learn a metric that assigns small (resp. large) distance to pairs of examples that are semantically similar (resp. dissimilar). This process of learning to map similar images in each other's vicinity is known an metric learning. To formalise this requirement, the loss will be defined over triplets of embeddings:

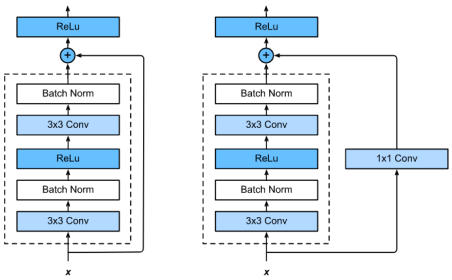

Fig. 3. The difference between a regular block (left) and a residual block (right). In the latter case, we can short-circuit the convolutions. (Source: d2l.ai)

1) an anchor (a)
2) a positive of the same class as the anchor (p)
3) a negative of a different class (n)

For some distance on the embedding space d , the loss of a triplet (a, p, n) is:

$$\mathcal{L} = max(d(a, p) - d(a, n) + margin, 0)$$

Goal is to minimize this loss, which pushes $d(a, p)$ to 0 and $d(a, n)$ to be greater than $d(a, p) + margin$. As soon as n becomes an "easy negative", the loss becomes zero.

Fig. 4. Triplet networks are trained using a triplet loss such that a metric space is learned where similar instances are close to each other, while dissimilar instances are far away from each other.

D. Triplet Mining

A crucial part of optimizing the triplet loss is in the selection stage of what set of triplets should be processed in each mini-batch. Based on the definition of the loss, there are three categories of triplets:

1) **easy triplets**: triplets which have a loss of 0, because $d(a, p) + margin < d(a, n)$. Negative in this instance is called easy negative.
2) **hard triplets**: triplets where the negative is closer to the anchor than the positive, i.e. $d(a, n) < d(a, p)$. Negative in this instance is called hard negative.
3) **semi-hard triplets**: triplets where the negative is not closer to the anchor than the positive, but which still have positive loss: $d(a, p) < d(a, n) < d(a, p) + margin$. Negative in this instance is called semi-hard negative.

Offline triplet mining

The first way to produce triplets is to find them offline, at the beginning of each epoch for instance. We compute all the embeddings on the training set, and then only select hard or semi-hard triplets. We can then train one epoch on these triplets.

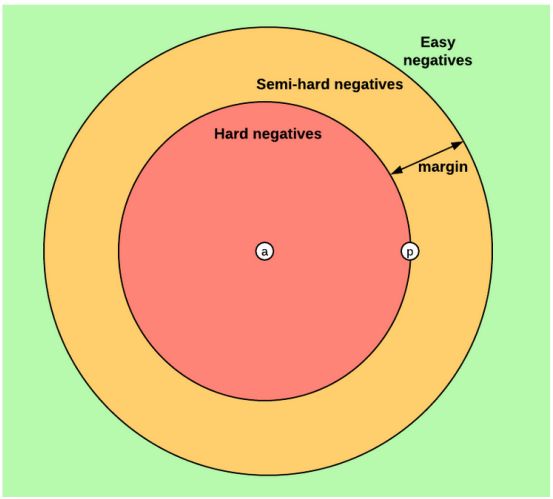

Fig. 5. Source: omoindrot.github.io

Concretely, we would produce a list of triplets (i, j, k). We would then create batches of these triplets of size B, which means we will have to compute 3B embeddings to get the B triplets, compute the loss of these B triplets and then backpropagate into the network. Overall this technique is not very efficient since we need to do a full pass on the training set to generate triplets. It also requires to update the offline mined triplets regularly. Fig explains.

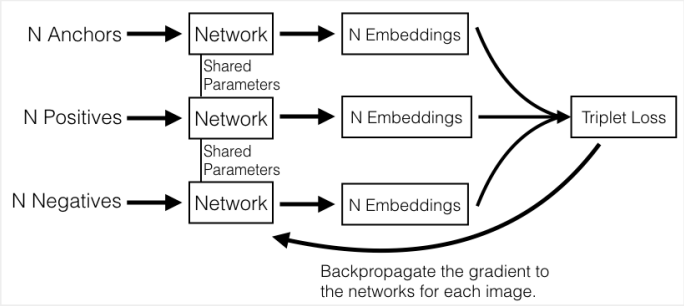

Fig. 6. Using three networks with shared parameters is a valid optimization approach, but inefficient because of compute and memory constraints. (Source: bamos.github.io)

Online Triplet Minining

The idea here is to compute useful triplets on the fly, for each batch of inputs. Given a batch of B examples (for instance B images of faces), we compute the B embeddings and we then can find a maximum of B^3 triplets. Of course, most of these triplets are not valid (i.e. they don't have 2 positives and 1 negative). Fig explains.

Strategies in online mining []

In online mining, we have computed a batch of B embeddings from a batch of B inputs. Now we want to generate triplets from these B embeddings.

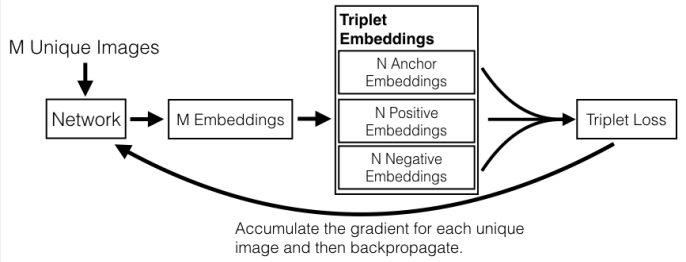

Fig. 7. Network doesn't have to be replicated with shared parameters and that instead a single network can be used on the unique images by mapping embeddings to triplets. (Source: bamos.github.io)

Whenever we have three indices $i, j, k \epsilon [1, B]$, if examples i and j have the same label but are distinct, and example k has a different label, we say that (i, j, k) is a valid triplet. What remains here is to have a good strategy to pick triplets among the valid ones on which to compute the loss.

They suppose that you have a batch of faces as input of size $B = PK$, composed of P different persons with K images each. A typical value is $K = 4$ The two strategies are:

1) **batch all**: select all the valid triplets, and average the loss on the hard and semi-hard triplets. A crucial point here is to not take into account the easy triplets (those with loss 0), as averaging on them would make the overall loss very small this produces a total of $PK(K1)(PKK)$ triplets (PK anchors, $K1$ possible positives per anchor, PKK possible negatives)
2) **batch hard**: for each anchor, select the hardest positive (biggest distance $d(a, p)$) and the hardest negative among the batch this produces PK triplets the selected triplets are the hardest among the batch.

The batch hard strategy yields the best performance. [4]

E. Optimizing

Gradient Descent with momentum or just Momentum speeds up the optimization of the cost function. It makes use of the moving average to update the trainable parameters of the neural network. Learning rate equal to 0.007 and momentum equal to 0.9. The optimizer schedules itself by updating the gamma every 10 epochs.

F. Testing

The class of an unseen data is known by k-Nearest Neighbour where:

1) Embeddings of Test dataset is projected on euclidean space trained encoder network.
2) Cosine similarity is found between test embeddings and train embeddings.
3) Classes of Test data is assigned by k-nearest train data embeddings.

IV. (

Experiments)

The model was tested only on Humpback Whale identification challenge in Kaggle.

TABLE I
TRAINING ON GTX 1080

Head	*ResNet Architecture*	*Dimensions*	*Time*	*Epochs*
1.	ResNet-50	1000	32minutes	15
2.	ResNet-50	500	32minutes	15
1.	ResNet-34	500	20minutes	15
1.	ResNet-18	500	11minutes	15

[a]Sample of a Table footnote.

A. Humpback Whale Challenge

The Humpback Whale Identification data set was collected from Happywhale.com. It consists 10000 of fluke images of different whales. Each whale is assigned a a class ID, and there are 5000 unique IDs. Before training, there was a series of affine transformations, cropping using bounding box method to extract only the relevant part of the image. The results of training process are showcased in Table 1. Training loss and Validation loss of different architectures are shared in following sections.

The model attained accuracy of 94% on ResNet50 architecture with 1000 dimensions.

REFERENCES

[1] Florian Schroff, Dmitry Kalenichenko, James Philbin: "FaceNet: A unified embedding for face recognition and clustering." CVPR 2015: 815-823
[2] Koch, G., Zemel, R. Salakhutdinov, R. (2015). Siamese Neural Networks for One-shot Image Recognition
[3] Hadsell, Raia, Sumit Chopra, and Yann LeCun. "Dimensionality reduction by learning an invariant mapping." null. IEEE, 2006.
[4] Alexander Hermans, Lucas Beyer, Bastian Leibe. "In Defense of the Triplet Loss for Person Re-Identification"

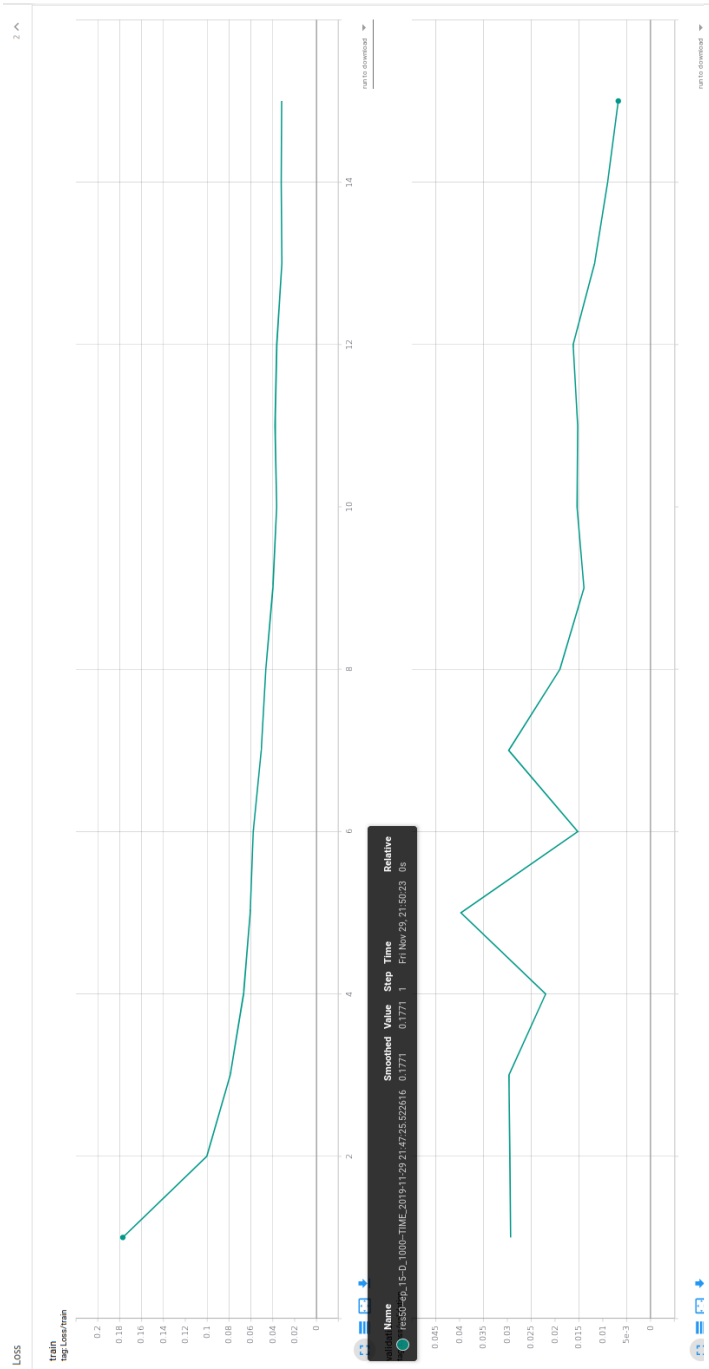

Fig. 8. Training and Validation loss for ResNet 50 architecture with 1000 Dimensions.

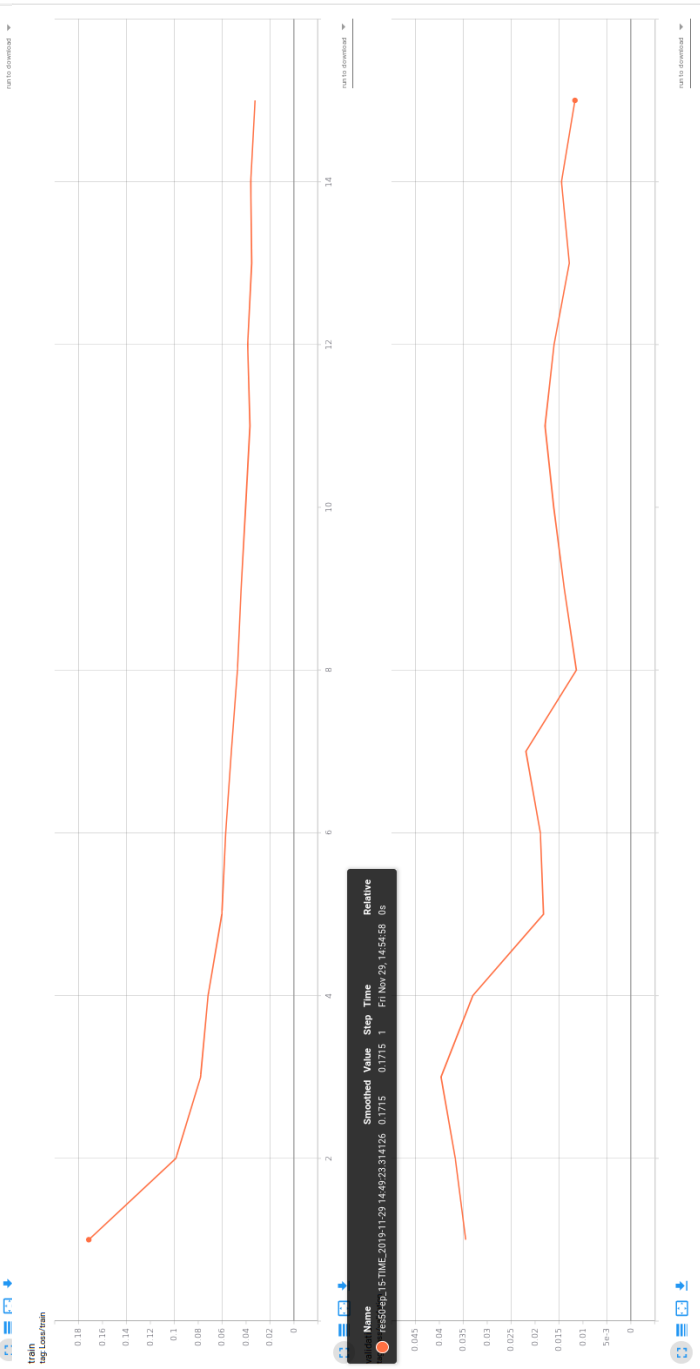

Fig. 9. Training and Validation loss for ResNet 50 architecture with 500 Dimensions.

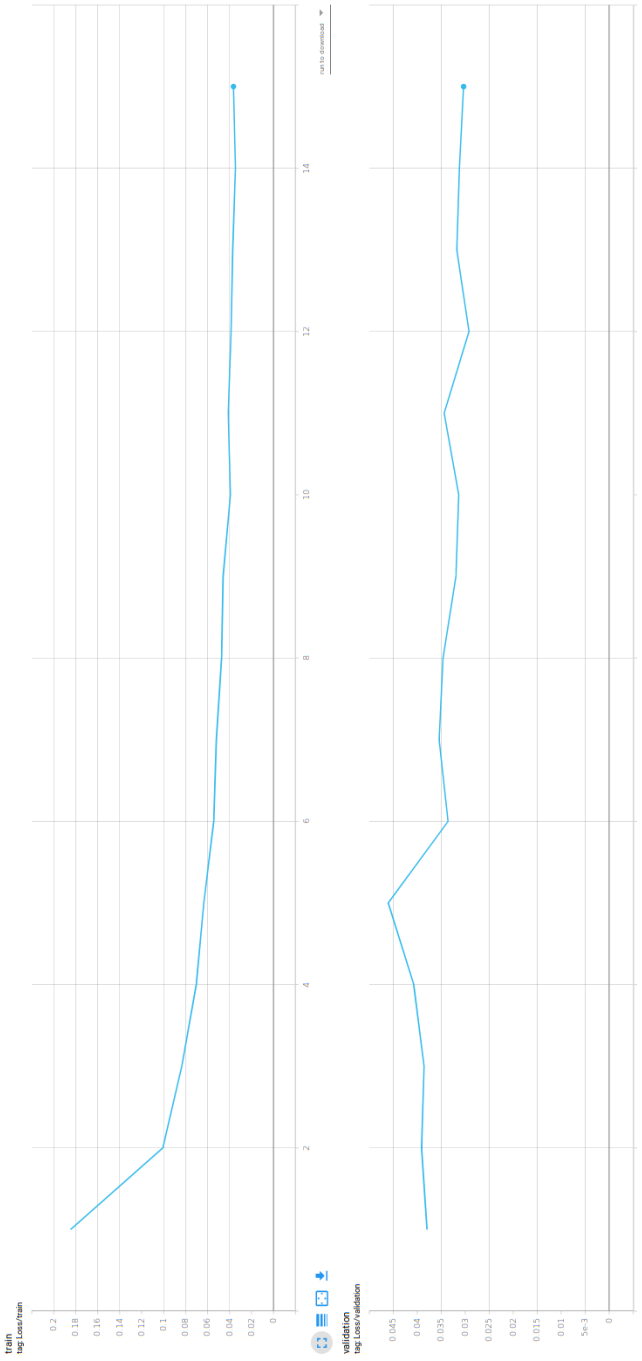

Fig. 10. Training and Validation loss for ResNet 34 architecture with 500 Dimensions.

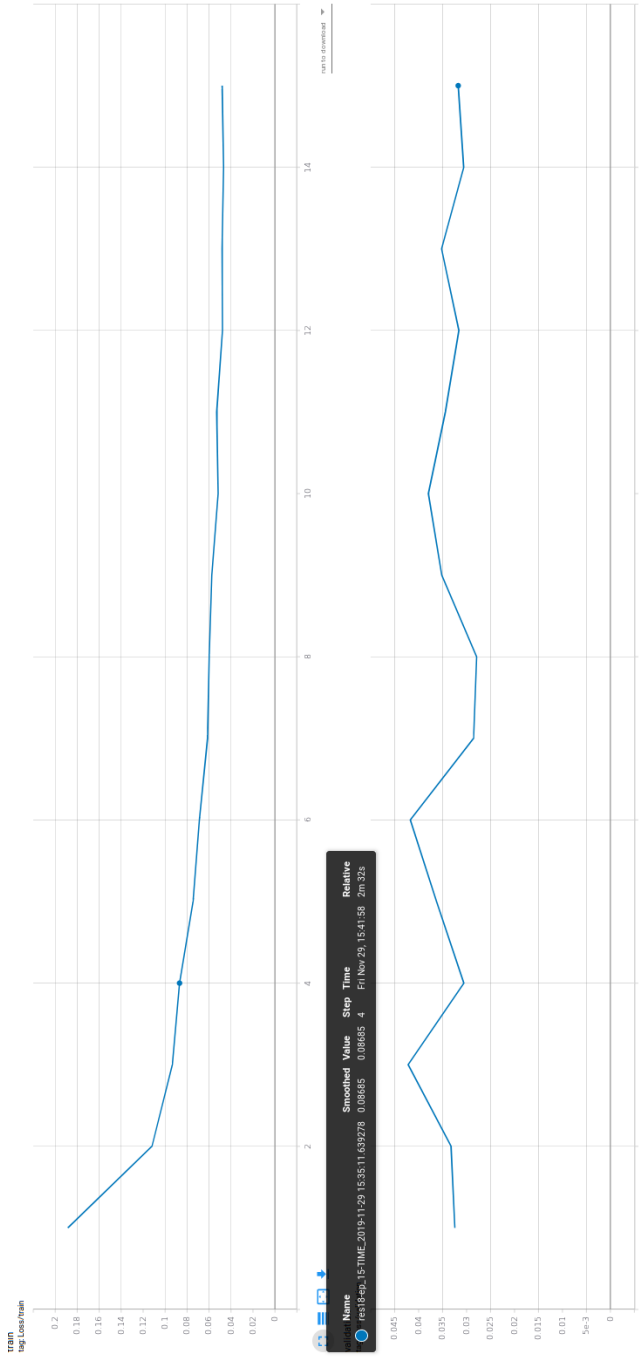

Fig. 11. Training and Validation loss for ResNet 18 architecture with 500 Dimensions.