Channel-Wise Early Stopping without a Validation Set via NNK Polytope Interpolation

David Bonet*, Antonio Ortega†, Javier Ruiz-Hidalgo*, Sarath Shekkizhar†
* Universitat Politècnica de Catalunya, Barcelona, Spain
† University of Southern California, Los Angeles, USA
email: davidbonetsole@gmail.com, aortega@usc.edu, j.ruiz@upc.edu, shekkizh@usc.edu

***Abstract*—State-of-the-art neural network architectures continue to scale in size and deliver impressive generalization results, although this comes at the expense of limited interpretability. In particular, a key challenge is to determine when to stop training the model, as this has a significant impact on generalization. Convolutional neural networks (ConvNets) comprise high-dimensional feature spaces formed by the aggregation of multiple channels, where analyzing intermediate data representations and the model's evolution can be challenging owing to the curse of dimensionality. We present channel-wise DeepNNK (CW-DeepNNK), a novel channel-wise generalization estimate based on non-negative kernel regression (NNK) graphs with which we perform local polytope interpolation on low-dimensional channels. This method leads to instance-based interpretability of both the learned data representations and the relationship between channels. Motivated by our observations, we use CW-DeepNNK to propose a novel early stopping criterion that (i) does not require a validation set, (ii) is based on a task performance metric, and (iii) allows stopping to be reached at different points for each channel. Our experiments demonstrate that our proposed method has advantages as compared to the standard criterion based on validation set performance.**

I. Introduction

Graphs play an important role in many machine learning applications and are used to model data structures and similarities in a dataset [1], [2]. The ability of graphs to define relationships between different types of entities allows us to describe and analyze complex patterns in data [3]. Recently, graphs have been used to understand and improve intermediate representations of deep learning models with application to various tasks, such as model regularization [4] and robustness [5], model distillation [6], and model interpretation [7], [8]. Since no graph is given a priori, these methods typically begin with a graph construction phase, where each graph node corresponds to an item in the training set, and the weight of an edge between two nodes is a function of the distance between their respective intermediate layer activations (i.e., their respective feature vectors). However, as deep learning models continue to grow in size, it is unclear if graphs constructed in these increasingly higher dimension spaces [9], [10] are able to capture relevant similarity information owing to the *curse of dimensionality*. Further, higher dimensions also increase the computational requirements for graph construction.

In this work, we use a novel graph construction and develop an improved model analysis for deep learning by taking advantage of a property of convolutional neural networks (ConvNets) [11], [12], [13]. Specifically, we note that convolutional layers (generally followed by non-linear and pooling layers) are formed by multiple convolutional filters that are applied in parallel to a common input (e.g., an original image or the output of the previous layer). The output of each of these is the aggregation of the outputs of multiple *channels*, where each channel corresponds to a single convolutional filter. Thus, in all ConvNet layers, except the last fully connected classification layers, high-dimensional intermediate layer feature vectors can be viewed as a concatenation of lower dimensional *subvectors*, each corresponding to the output of a channel. We make use of this natural "subvectorization" in ConvNets to develop a model analysis through channel-wise graph constructions as an alternative to the standard full layer model analysis [14], [15], [16]. This also allows us to estimate model generalization at the channel level.

In particular, we propose a channel-wise extension of DeepNNK [7], CW-DeepNNK, which leverages the geometrical interpretation and the robustness of NNK [10] to obtain per channel leave one out (LOO) classification estimates. Our analysis allows us to show that each channel learns specific features, each having different levels of importance for the overall task. Moreover, features extracted by each channel complement each other, so that the combination of all subvectors leads to a better classification than that achieved by each individual channel. A more detailed study of the properties of channel-wise graphs is carried out in [17], where the intrinsic dimension of the data representations at a subvector level and data relationships between channels are studied.

We also conjecture that learning happens at different rates in each channel and propose a channel-wise early stopping criterion for deep learning systems that does not require any validation data. Training is stopped for a given channel if its corresponding LOO classification performance no longer improves. Our proposed method achieves comparable test performance with fewer training iterations than existing methods, such as validation-set-based early stopping and aggregate feature vector based DeepNNK [7]. Our early stopping criterion requires lower runtimes and has the further advantage of not requiring any validation data to be set aside. We present strategies to further reduce the complexity of our algorithm while being able to maintain a good estimate of model generalization. Our framework is compatible with any optimizer and can be integrated with any ConvNet training setting, while

not requiring hyperparameter tuning. This method will be particularly useful for problems with small datasets, where holding out data for validation is impractical.

II. Background and Related work

A. Notation

We denote scalars, vectors, random variables, and matrices using lowercase (e.g., x and θ), lowercase bold (e.g., $\boldsymbol{x}$ and $\boldsymbol{\theta}$), uppercase (e.g., X and Y), and uppercase bold (e.g., $\boldsymbol{K}$ and $\boldsymbol{\Phi}$) letters, respectively. Wherever applicable, we use a superscript to index and denote a subvector and related measures. For example, a vector $\boldsymbol{x}$ in $\mathbb{R}^D$ obtained as the concatenation of S subvectors $\boldsymbol{x}_i^s \in \mathbb{R}^{D_s}$ where $\sum_{s=1}^{S} D_s = D$:

$$\boldsymbol{x}_i = \begin{bmatrix} \boldsymbol{x}_i^1 \\ \boldsymbol{x}_i^2 \\ \vdots \\ \boldsymbol{x}_i^S \end{bmatrix} \in \mathbb{R}^D \tag{1}$$

$\mathcal{D}_{\text{train}} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2) \dots (\boldsymbol{x}_N, y_N)\}$ is the set of training data and $\mathcal{D}_{\text{train}}^i$ is the set obtained by removing $(\boldsymbol{x}_i, y_i)$ from $\mathcal{D}_{\text{train}}$. We denote the empirical risk or generalization error associated with a function $\hat{f}$ on M data points $\mathcal{D}$ as

$$\mathcal{R}_{\text{emp}}(\hat{f}|\mathcal{D}) = \frac{1}{M} \sum_i l(\hat{f}(\boldsymbol{x}_i), y_i) \tag{2}$$

where $\hat{f}(\boldsymbol{x})$ is the prediction at $\boldsymbol{x}$ and $l(\hat{f}(\boldsymbol{x}_i), y_i)$ is the error in the estimate at $\boldsymbol{x}_i$ relative to y_i.

Given the training data, the leave one out (LOO) [18] estimate of a function at $\boldsymbol{x}_i$ is the estimate based on the set containing all training points except $\boldsymbol{x}_i$. We denote the risk associated with the LOO procedure as

$$\mathcal{R}_{\text{LOO}}(\hat{f}|\mathcal{D}_{\text{train}}) = \frac{1}{N} \sum_{i=1}^{N} l(\hat{f}(\boldsymbol{x}_i)|\mathcal{D}_{\text{train}}^i, y_i). \tag{3}$$

B. Non Negative Kernel (NNK) regression graphs

Given N data points represented by feature vectors $\boldsymbol{x}$, a graph is constructed by connecting each data point (node) to similar data points, so that the weight of an edge between two nodes is based on the similarity of the data points, with the absence of an edge (a zero weight) denoting least similarity. NNK assumes local smoothness in a non linearly transformed space, the Reproducing Kernel Hilbert Space (RKHS) [19], corresponding to positive definite kernels used to measure similarity between data points, such as the Gaussian kernel with bandwidth σ of (4) or the range normalized cosine kernel of (5).

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2 / 2\sigma^2\right) \tag{4}$$

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = 1/2 + \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle / \left(2\|\boldsymbol{x}_i\|\|\boldsymbol{x}_j\|\right) \tag{5}$$

Unlike weighted K-nearest neighbor (KNN) [20] and ϵ-neighborhood graphs (ϵ-graphs) [21] that are sensitive to the choice of hyperparameters K/ϵ, non negative kernel regression (NNK) graphs [10] are suggested as a principled approach to graph construction based on a signal representation view.

While KNN is still used as an initialization, NNK performs a further optimization akin to orthogonal matching pursuit [22] in kernel space, resulting in a *robust* representation with the added advantage of having a *geometric* interpretation. The Kernel Ratio Interval (KRI) theorem in [10] reduces the local NNK graph construction problem (deciding which of the KNN neighbors of a given node should also be NNK neighbors) into a series of hyper plane conditions, one per NNK weighted neighbor, which applied inductively lead to a convex polytope around each data point, as illustrated in Figure 1 (NNK graph on the right). Intuitively, NNK ignores data that are further away along a *similar* direction as an already chosen point and looks for neighbors in an *orthogonal* direction.

C. DeepNNK: Neural Networks and NNK interpolation

DeepNNK [7] is a non-parametric interpolation framework based on local polytopes obtained using NNK graphs [10] that replaces the standard softmax classification layer of a neural network using the activations obtained at the penultimate layer as input features, for model evaluation and inference. A key advantage of DeepNNK lies in the fact that label interpolation is performed based on the relative positions of points in the training set, which makes it possible to perform leave one out estimation to characterize the task performance of a model and its generalization without the need for additional data, namely the validation set. Note that this would not be possible in the original configuration where classification is performed based on a parametric class boundary.

Formally, the DeepNNK interpolation estimate at a data point $\boldsymbol{x}$ is given by

$$\hat{f}_{\text{NNK}}(\boldsymbol{x}) = \sum_{i \in \text{NNK}_{\text{poly}}(\boldsymbol{x})} \frac{\theta_i \, y_i}{\sum_{j \in \text{NNK}_{\text{poly}}(\boldsymbol{x})} \theta_j} \tag{6}$$

where $\text{NNK}_{\text{poly}}(\boldsymbol{x})$ is the set of indices of NNK neighbors that form a convex polytope around $\boldsymbol{x}$ and $\boldsymbol{\theta}$ corresponds to the NNK weights associated with these neighbors. Note that in DeepNNK the standard kernels are used to estimate similarity between data points after the non-linear mapping $\boldsymbol{h}$, corresponding to a deep neural network (DNN), has been applied. For example, the Gaussian kernel of (4) is rewritten as

$$k_{\text{DNN}}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\|\boldsymbol{h}(\boldsymbol{x}_i) - \boldsymbol{h}(\boldsymbol{x}_j)\|^2 / 2\sigma^2\right) \tag{7}$$

The authors of [7] show that NNK graphs outperform weighted KNN graphs in label interpolation and that $\mathcal{R}_{\text{LOO}}(\hat{f}_{\text{NNK}}|\mathcal{D}_{\text{train}})$ can be a better indicator of generalization than the $\mathcal{R}_{\text{emp}}(\hat{f}_{\text{DNN}}|\mathcal{D}_{\text{train}})$ associated with a DNN classifier applied on training data.

D. Early stopping methods

The central idea behind early stopping (stop training or optimization) [23] is that there exists a critical regime during the training of a learning model where the model ceases to generalize (perform better) on unseen data points while being able to do improve performance on given training data. Identifying this point of negative or zero return is also

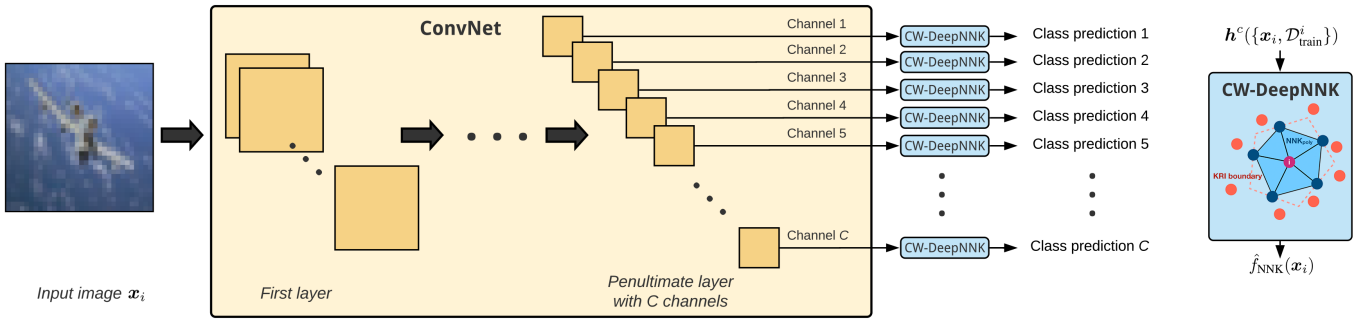


Fig. 1: CW-DeepNNK interpolation framework integrated in a ConvNet, replacing the last softmax classification layer. Taking as inputs the activations from individual channels in the penultimate layer, CW-DeepNNK outputs C class predictions for a given input $\boldsymbol{x}_i$ via NNK polytope interpolation. Application of the LOO procedure results in C channel-wise generalization estimates which can be used to construct an early stopping criterion without a validation set.

attractive from a computational perspective and is the goal of various early stopping rules or methods in machine learning [24], [25], [26]. A conventional and widely popular early stopping method in machine learning is the one based on validation data, which we name as *Validation-based* method. Here, one sets aside a part of the training set (referred to as validation set) not to be used for training the model, but on which one only evaluates the performance of the trained model. The validation performance is taken as a proxy for model generalization with training halted when the model begins to perform poorly on the validation set.

Although very effective in practice, especially with large training datasets where holding off a small part of the training data has no effect in the learning process, there are drawbacks to Validation-based early stopping [26]. The validation performance may have a large stochastic error depending on the size of the validation set and may introduce biases leading to poor generalization estimates. A large validation set yields a more robust generalization estimate but may deprive the model of valuable information by reducing significantly the amount of data available for training. Thus, in applications with limited training data, it may be better to use all of it for training, rather than holding some of it for validation.

This shortcoming of Validation-based early stopping has led to recent alternatives such as [27], [26]. First, [27] proposed a stopping rule based on estimating the marginal likelihood by tracking the change in entropy of the posterior distribution of the network parameters as an indicator of generalization. However, the likelihood estimates in this framework are affected when the model has additional regularization terms, which is typically the case for most state-of-the-art methods. An alternative approach [26] presents an early stopping method based on a fast-to-compute local gradient statistic. This method obtains good results compared to the Validation-based method, but requires hyperparameter tuning. Moreover, both [27] and [26] rely on gradient-related statistics that are only valid in standard stochastic gradient descent settings and fail to generalize to more advanced optimizers, such as those based on momentum. Due to their limitations, these methods have been not widely used in practice. A more detailed comparison with our proposed method is carried out in Section IV-C.

III. Channel-Wise DeepNNK (CW-DeepNNK)

DeepNNK [7] aims at estimating generalization error with the LOO procedure. However, it fails to utilize the structure of the feature vector obtained as the concatenation of different channels. The data used for interpolation is often very high-dimensional, which can lead to poor representation of the "true" similarity between data points and therefore to sub-optimal performance of polytope-based label interpolation of (6). Partially due to this, estimation of generalization performance (on an unseen dataset) using (6) does not achieve the same accuracy as the conventional procedure based on using a separate validation set.

To address this problem, in this work we propose a local polytope label interpolation in individual channels (CW-DeepNNK), as illustrated in Figure 1. Instead of using the transformed data representations of the full penultimate layer $\boldsymbol{h}(\boldsymbol{x})$, which consists of the aggregation of outputs of C convolutional channels, we propose dividing the feature space into channels:

$$\boldsymbol{h}(\boldsymbol{x}_i) = \begin{bmatrix} \boldsymbol{h}^1(\boldsymbol{x}_i) \\ \boldsymbol{h}^2(\boldsymbol{x}_i) \\ \vdots \\ \boldsymbol{h}^C(\boldsymbol{x}_i) \end{bmatrix} \in \mathbb{R}^{D_{\boldsymbol{h}}}, \tag{8}$$

which are well-defined and can be interpreted individually. Then, in each channel c, the first step for the CW-DeepNNK LOO procedure is to use the intermediate representations $\boldsymbol{h}^c(\mathcal{D}^i_{\text{train}})$ as feature vectors to construct an NNK neighborhood for each data point $\boldsymbol{h}^c(\boldsymbol{x}_i)$ in the training set. Second, perform the NNK interpolation (6). Finally, we compute the LOO estimation (3) per channel, obtaining the CW-DeepNNK label interpolation errors $\mathcal{R}^1_{\text{LOO}}, \mathcal{R}^2_{\text{LOO}}, \ldots, \mathcal{R}^C_{\text{LOO}}$.

By using the CW-DeepNNK procedure at each training epoch we obtain a label interpolation error curve for each channel, which allow us to monitor the generalization of the model during training. Then, we propose a novel channel-wise early stopping criterion described in Algorithm 1, which does not require a validation set and the stopping is performed in stages, allowing us to stop the training of each channel independently. Note that this is the first early stopping criterion that (i) does not require a separate validation set and (ii)

is based on a task performance metric (e.g., accuracy). In contrast, the other proposed early stopping methods without a validation set [26], [27] based their generalization estimation on properties of the gradients computed during training.

Algorithm 1 CW-DeepNNK progressive early stopping without a validation set

Input:

$\mathcal{D}_{\text{train}} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2) \dots (\boldsymbol{x}_N, y_N)\}$: training set
$\boldsymbol{w}$: model parameters $\triangleright$ $\boldsymbol{w}_{\text{penult}}$: penultimate layer params
$\boldsymbol{h}$: model non-linear mapping at the penultimate layer
C: number of channels in penultimate layer
n: number of steps between generalization evaluations
p: *patience*, number of times to observe worsening LOO NNK interpolation error before stopping channel
K: number of initial neighbors

Output: best parameters $\boldsymbol{w}^*$, best number of training steps t^*

$t = 0$, $\boldsymbol{q} = \boldsymbol{p}$, $\boldsymbol{r} = \boldsymbol{\infty}$
initialize $\boldsymbol{w}$
$\boldsymbol{w}^* = \boldsymbol{w}$, $t^* = t$
while $\boldsymbol{q} \neq \boldsymbol{0}$ **do**
  Update $\boldsymbol{w}$ by running the training algorithm for n steps
  $t = t + n$
  for $c = 1 : C$ **do**
    if $\boldsymbol{q}(c) > 0$ **then**
      for $i = 1 : N$ **do**
        $\boldsymbol{\theta} = \textit{findNNKneighbors}(\boldsymbol{h}^c(\{\boldsymbol{x}_i, \mathcal{D}^i_{\text{train}}\}), K)$
        $\hat{f}_{\text{NNK}}(\boldsymbol{x}_i) = \sum_j \frac{\theta_j y_j}{\sum_l \theta_l}$
      end for
      $\mathcal{R}^c_{\text{LOO}} = \frac{1}{N} \sum_i l(\hat{f}_{\text{NNK}}(\boldsymbol{x}_i) | \mathcal{D}^i_{\text{train}}, y_i)$
      if $\mathcal{R}^c_{\text{LOO}} < \boldsymbol{r}(c)$ **then**
        $\boldsymbol{r}(c) = \mathcal{R}^c_{\text{LOO}}$, $\boldsymbol{q}(c) = p$
        $\boldsymbol{w}^* = \boldsymbol{w}$, $t^* = t$
      else
        $\boldsymbol{q}(c) = \boldsymbol{q}(c) - 1$
      end if
      if $\boldsymbol{q}(c) = 0$ **then**
        Freeze and stop training $\boldsymbol{w}^c_{\text{penult}}$
      end if
    end if
  end for
end while

A. Channel-wise early stopping without a validation set

Starting from the standard *patience* criterion, we monitor the generalization performance in the penultimate layer channels and we use a patience parameter p in each channel. When a channel stops generalizing, i.e., $\mathcal{R}^c_{\text{LOO}}$ has not improved in p observations, we freeze the model parameters of the channel and stop training it. The rest of the model continues learning until each of the channels stops generalizing, where we consider that we have reached the optimal point and the overall generalization of the model no longer improves. Finally, we save the best model parameters $\boldsymbol{w}^*$ where the last minimum generalization error is detected. Code for the proposed method is available online.[1]

B. Complexity

The channel-wise LOO label interpolation *baseline* is computationally expensive, since it requires constructing one NNK graph per training instance, per channel, and per epoch. However, several improvements can be made to achieve an overall lower computation cost, competitive with the state-of-the-art early stopping methods in large scale problems. First, performing NNK independently in every epoch is very costly, but the features learned in a given channel in consecutive epochs are similar. Thus, complexity can be reduced by a factor of T by performing one LOO estimation every T epochs. As an alternative, we can maintain a fixed NNK assignment for all nodes and monitor the approximation error for node i as a function of its neighbors. When this error increases a new NNK graph is constructed, so that a better set of neighbors for node i can be found.

Other improvements in efficiency can be achieved by reducing the number of channels and the number of nodes for which LOO label interpolation is performed. For example, we could perform the interpolation only on those channels with lowest error. We could also perform random subsampling of training data points instead of performing the full LOO procedure. As a result, computing time would be drastically reduced, at the expense of less reliable and more random generalization estimates, whose impact could be reduced by augmenting the patience parameter, which would make the actual stopping point less sensitive to random oscillations in the estimates.

Additionally, while selecting K for the experiments we observed that both NNK-based methods are very robust with respect to the selection of this single hyperparameter. Therefore, K could be chosen as the minimum K that yields stable results, significantly reducing the complexity. Exploring all these ideas for further efficiency improvement of the proposed method is left for future work.

IV. Experiments

In this section, we evaluate CW-DeepNNK for estimating channel generalization and construct an efficient early stopping method with several benefits over DeepNNK and the Validation-based method. We focus our experiments on a binary classification setting using 2 classes of CIFAR-10 dataset [28]: "plane" and "ship". Training set and validation set, if needed, are split in a stratified fashion in each experiment ensuring class balance in each set. We consider a 7 layer ConvNet architecture consisting of 4 convolution layers with 5 depth channels and ReLU activations, 2 max-pool layers and a fully connected softmax layer. We train the model with the Adam optimizer [29], with a learning rate of 0.001 and batch size of 50 on a GTX Titan X with 8GB of memory. We compare the model performance (based on softmax classification) with that of DeepNNK and CW-DeepNNK interpolation on train data. We observe that some

[1] https://github.com/STAC-USC/CW-DeepNNK_Early_Stopping

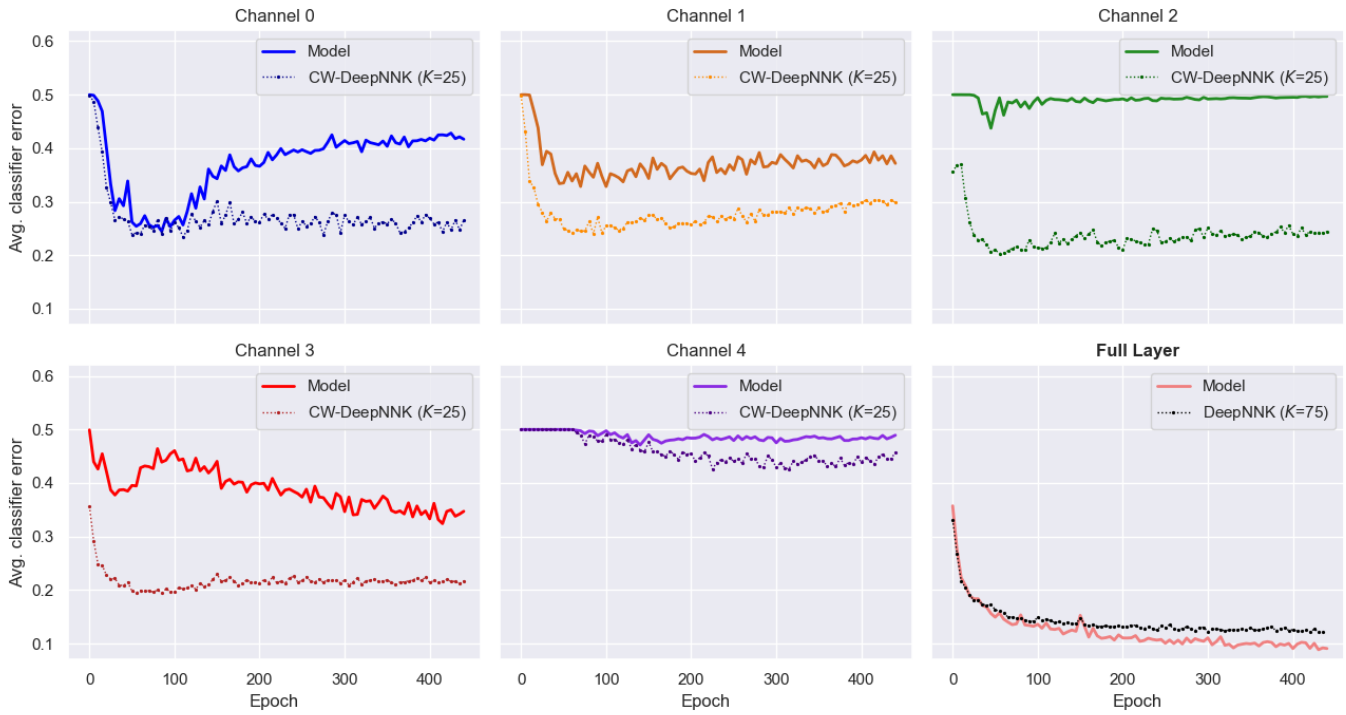


Fig. 2: Classification error using interpolating classifiers (DeepNNK, CW-DeepNNK) and model error on training data. CW-DeepNNK significantly outperforms the softmax classification of the neural network with only a single channel of the penultimate layer activated. Further, most channels attain their CW-DeepNNK minima at an early number of iterations with performance deteriorating later indicative of overfitting.

channels learn more valuable features than others for the classification task (see Figure 2). We analyze the behaviour of our generalization estimate in the presence and absence of dropout regularization in Section IV-B and in Section IV-C compare NNK-based generalization estimates with that of the Validation-based strategy [23] to perform early stopping and discuss the complexity of each.

A. *Generalization estimates using CW-DeepNNK*

Each channel of a convolutional layer defines a feature subspace where we should be able to quantify how useful the information from that channel is for the classification task, and have a better interpretation of the captured features in each channel.

Figure 2 shows a comparison between model error on training data, DeepNNK and CW-DeepNNK label interpolation error with LOO estimation. In the DeepNNK case, the error gap between the model on train data and LOO DeepNNK increases with the epochs, indicating that the generalization performance is worsening and the model starts to overfit to the train data. Also, we can see how CW-DeepNNK has a much better performance than the model when only a single channel is activated. Note that an error of 0.5 in a binary classification is as bad as doing random classification. We also observe how the interpolation error in the channels soon reaches a minimum, and then the classification error increases again. This minimum may indicate the optimal point of generalization in each channel, from which the learned features begin to overfit the training data.

Although the last fully connected layer of the model is trained to use the full combination of features from all channels, we wanted to see what happens if the model has to perform classification when relying only on partial information. Figure 2 shows how our method is able to perform much better in each independent channel subspace, and the model is not capable of performing at a decent level when some feature channels are deactivated. Thus, we now have a new point of view, where we can estimate properly how useful each of the individual channels is for the task.

B. *Regularized vs. Non-Regularized models*

We can analyze the performance of CW-DeepNNK in different scenarios and compare the results in Section IV-A, with no regularization, to results obtained using explicit regularization, adding a dropout layer [30] after each convolutional layer, with a dropout probability of 0.2. In both cases we see individual channels that fail to learn features of the data relevant to the task, obtaining almost maximum error in binary classification $\mathcal{R}_{\text{LOO}} \approx 0.5$. In the following study we focus on the channels with an error $\mathcal{R}_{\text{LOO}} < 0.4$.

Figure 3 shows how in a non-regularized model, our CW-DeepNNK generalization error estimate finds a minimum at an early number of iterations. If we compare it with the test performance, we see that channel-wise LOO performances peak at a similar place to the peak of the test loss, where early stopping would occur using the Validation-based method. In

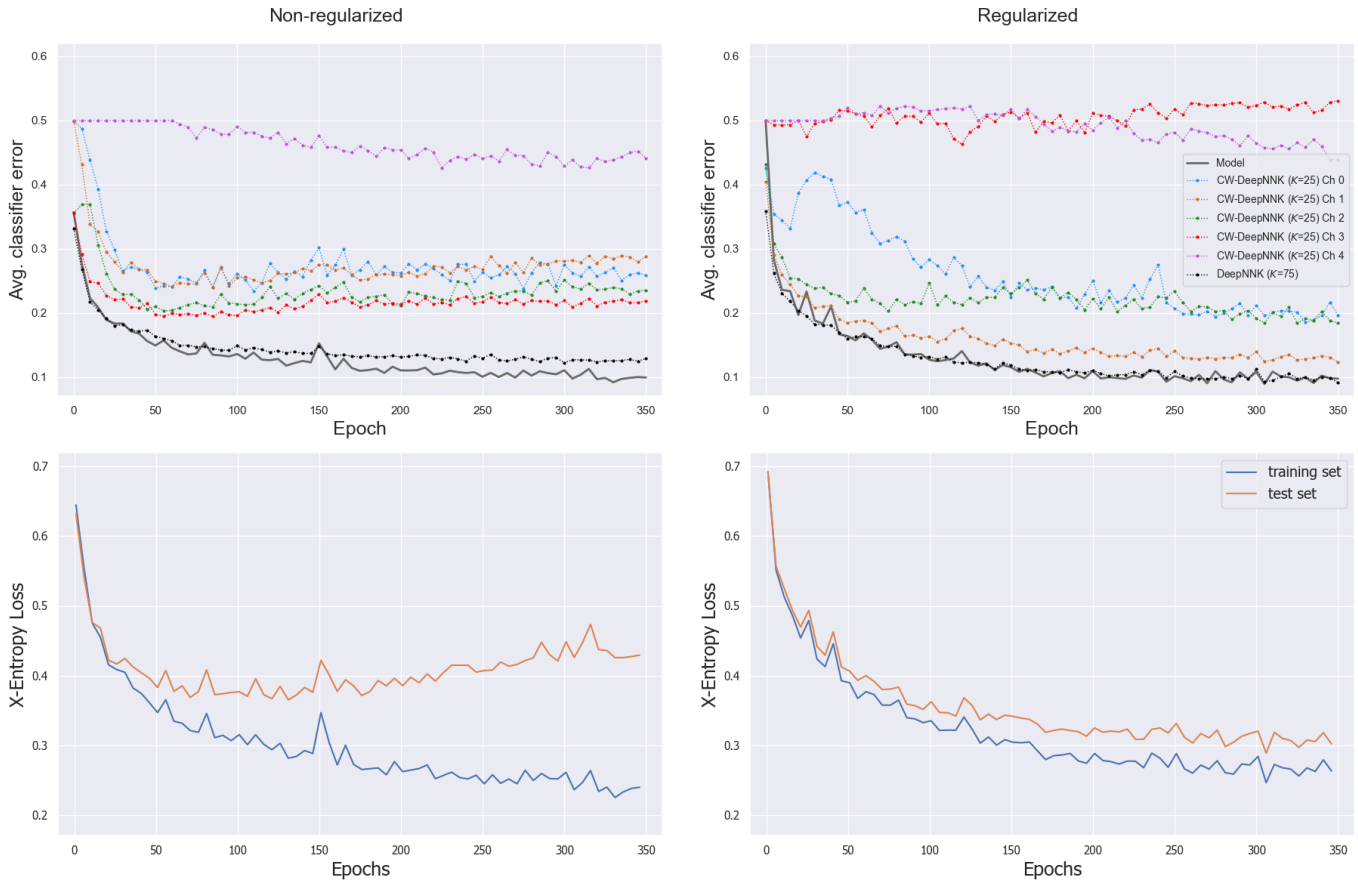


Fig. 3: **top row**: DeepNNK, CW-DeepNNK and training model error; **bottom row**: model loss on train and test data, metrics shown every 5 epochs. **left column**: no regularization; **right column**: dropout with a rate of 0.2 in each layer. With no regularization, most CW-DeepNNK minima coincide with that of the test loss, where early stopping should occur. In a well-regularized model, CW-DeepNNK continues improving as does the model in the test set.

addition, channel minimum error detection in different points in time suggests that early stopping could be performed progressively by channels. Further, monitoring this generalization estimator in a well-regularized network that does not overfit, our estimator is consistent, observing a performance curve similar to that of the test set.

We can also detect that in the non-regularized case, the *important* channels, i.e., those channels where best CW-DeepNNK performances are achieved, have a very similar but very poor performance with an error between 0.2 and 0.3. Instead, in the regularized case we can see that the important channels reach an error below 0.2.

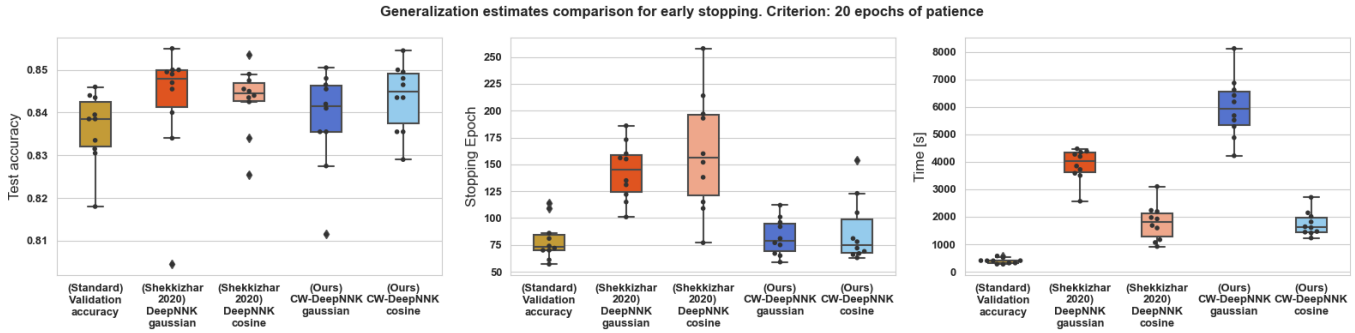
We note that only some channels of a convolutional layer are key for to the classification task, and that we can detect these channels even before performing the LOO interpolation based on the NNK polytope local geometry (number of NNK neighbors) and zero patterns in the activations.

We observe that in channels with fewer zero dimensions, we will obtain a higher dimensional NNK graph with higher same-class weights, which subsequently leads to better interpolation. Besides, the non-regularized networks have a more homogeneous behavior between channels but with worse performance in general, while for regularized models, the most important channels drive a better overall result.

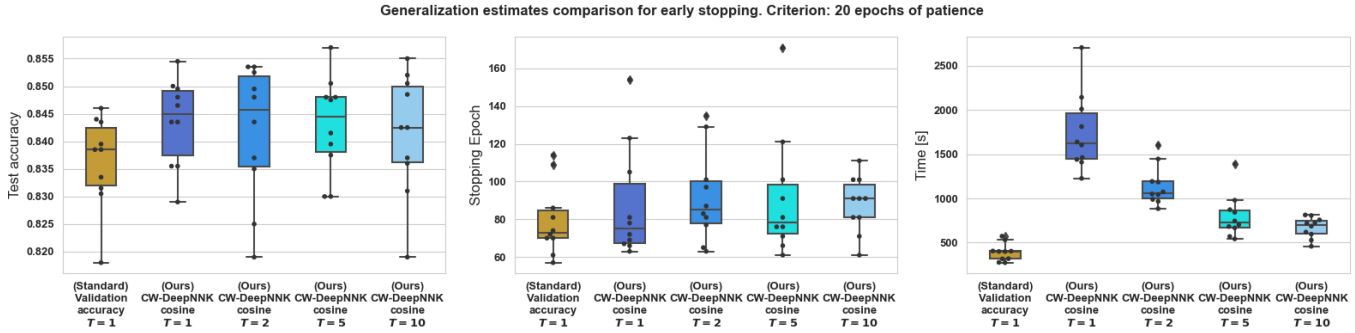
C. *Early stopping with CW-DeepNNK*

We compare our generalization estimate with the Validation-based method as well as with the DeepNNK estimate [7], to perform early stopping. In this case, we use a patience parameter as stopping rule for the Validation-based method and for the full layer LOO DeepNNK interpolation. For our estimate, we use a patience stopping rule in each channel, as described in Section III-A. Other non-validation set methods based on gradient-related statistics [26], [27] are promising approaches, but unlike our proposed method based on a task performance metric, they are not compatible with momentum-based optimizers and require hyperparameter tuning for a reliable stopping. Moreover, [26] assumes that all weights of a layer will converge at similar speeds, which may not be necessarily true in convolutional layers with multiple channels. The criterion proposed in [27] has only been validated on simpler models, e.g., one hidden layer networks.

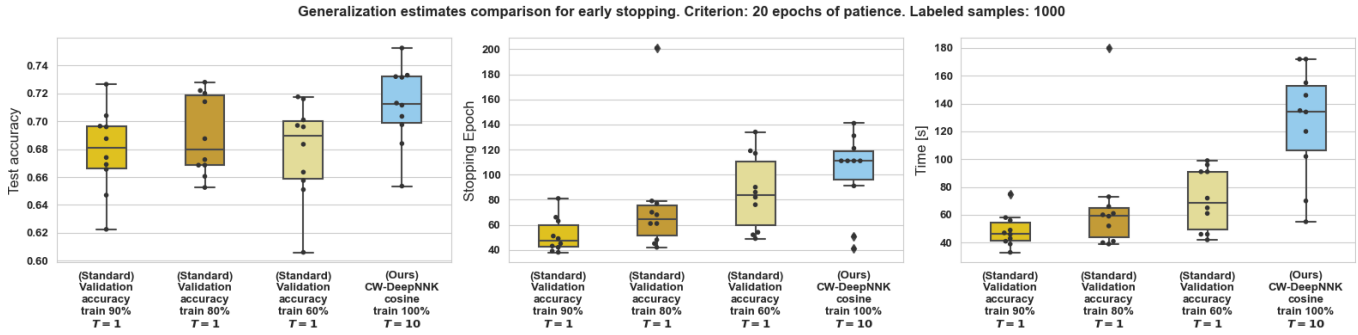
Note that other stopping rules [23], [31] could use our generalization estimate as a substitute for the validation curve, but we leave these experiments for future work. Even though weights of the penultimate layer are prevented from further training in our proposed channel-wise method, this does not result in extra benefits in computing complexity as the fraction



(a) Validation-based method using 20% of labeled samples as validation set, and DeepNNK and CW-DeepNNK for both Gaussian kernel (4) and cosine kernel (5). NNK-based methods obtain higher test accuracies than the Validation-based method, training the model with all labeled data. CW-DeepNNK cosine obtains the best trade-off between test accuracy and training iterations with an overhead of computational time.



(b) Comparison of LOO computation frequencies for CW-DeepNNK; Validation-based method using 20% of labeled samples as validation set. CW-DeepNNK frequency reduction preserves superior results while requiring less computation time, comparable to the Validation-based method.



(c) Small dataset case with 1000 labeled samples, obtained from random sampling at each initialization. CW-DeepNNK obtains higher accuracies than the Validation-based method, using all labeled data available for training.

Fig. 4: Test accuracy, stopping epoch and training elapsed time for different early stopping methods. Note that test accuracy refers to the accuracy obtained in the test set with the best model according to each criterion, i.e., where we find the last minimum generalization error, not the last version where we stop training. $K = 15$ for NNK-based methods.

of parameters that we stop training in intermediate stages is not significant. An extension of this method to the rest of the layers of the model could speed up the training iterations.

Figure 4a shows the results obtained using the different generalization estimates for early stopping, with a patience of 20 epochs and 10 different model initializations and validation set partitions. We can see how all NNK-based methods obtain test accuracies higher than the Validation-based method. Although the best models are obtained in the DeepNNK case, it requires a lot of epochs to find the optimal stopping point whereas the other methods are able to detect overfitting in a much earlier stage. In the two proposed methods based on NNK interpolation, using a patience parameter that is too small can lead to not finding the global minimum of generalization error, leading to premature stopping and lower test accuracy, as in the case of the outliers. Therefore, choosing a higher patience can ensure a high test accuracy, but at the cost of a longer training, often unnecessary. The proposed channel-wise generalization estimate using the range normalized cosine kernel (5) is the alternative that obtains the best trade-off between performance, training iterations and time. In Figure 4b we study different LOO computation frequencies to further improve the efficiency of the proposed algorithm, maintaining a total wait of 20 epochs in all cases. Test accuracy and stopping epochs are preserved while the computation time is significantly reduced with increasing T, reaching an efficiency

and results similar to those of the Validation-based method in the case of $T = 10$.

We also study the case of dealing with small datasets, where using the available data wisely is critical for good generalization and test performance. Using a labeled data subset of 1000 samples of the two CIFAR-10 classes (split between train set and validation set if required) in Figure 4c we see how our method obtains state-of-the-art results outperforming the standard Validation-based approach, since we can train the model with all labeled data without the need of separating data for a validation set.

V. Conclusion and Future work

We introduced a novel approach for channel-wise generalization estimation in ConvNets based on local NNK polytope interpolation, which allows to detect the channels that are the most important for the task and obtain best interpolation performance. We also presented a progressive channel-based early stopping strategy which does not require a validation set. Our approach may be the preferred for early stopping in situations where test performance is key or when labeled data is scarce, since we would not need to hold out data for a validation set. Based on a task performance metric, as is the widely used Validation-based method, the presented criterion can be integrated into any training setting (including momentum-based optimizers and explicit regularization methods) and does not require hyperparameter tuning.

Future work should explore the ideas presented to further improve efficiency, obtaining competitive results on large-scale problems. Future research could also be in the direction of neural network pruning based on the CW-DeepNNK interpolation error, since we have observed that in certain channels we have practically no useful information to perform the interpolation, and those channels could be pruned, resulting in a more compact network with less computational cost. Another line of research could involve a *progressive* early stopping of the full model, achieving a significant saving of gradient computation and backpropagation throughout the training.

References

[1] A. Ortega, P. Frossard, J. Kovačević, J. M. Moura, and P. Vandergheynst, "Graph signal processing: Overview, challenges, and applications," *Proceedings of the IEEE*, vol. 106, no. 5, pp. 808–828, 2018.

[2] A. Ortega, *Introduction to Graph Signal Processing*. Cambridge University Press, 2021.

[3] D. Bacciu, F. Errica, A. Micheli, and M. Podda, "A gentle introduction to deep learning for graphs," *Neural Networks*, 2020.

[4] C. Lassance, V. Gripon, and A. Ortega, "Laplacian networks: Bounding indicator function smoothness for neural networks robustness," *APSIPA Transactions on Signal and Information Processing*, vol. 10, 2021.

[5] C. Lassance, V. Gripon, and A. Ortega, "Representing deep neural networks latent space geometries with graphs," *Algorithms*, vol. 14, no. 2, p. 39, 2021.

[6] C. Lassance, M. Bontonou, G. B. Hacene, V. Gripon, J. Tang, and A. Ortega, "Deep geometric knowledge distillation with graphs," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8484–8488, IEEE, 2020.

[7] S. Shekkizhar and A. Ortega, "Deepnnk: Explaining deep models and their generalization using polytope interpolation," *arXiv preprint arXiv:2007.10505*, 2020.

[8] V. Gripon, A. Ortega, and B. Girault, "An inside look at deep neural networks using graph signal processing," in *2018 Information Theory and Applications Workshop (ITA)*, pp. 1–9, IEEE, 2018.

[9] N. Papernot and P. McDaniel, "Deep k-nearest neighbors: Towards confident, interpretable and robust deep learning," *arXiv preprint arXiv:1803.04765*, 2018.

[10] S. Shekkizhar and A. Ortega, "Graph construction from data by non-negative kernel regression," in *2020 IEEE Int. Conf. on Acoustics, Speech and Signal Process. (ICASSP)*, pp. 3892–3896, IEEE, 2020.

[11] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[12] S. Zagoruyko and N. Komodakis, "Wide residual networks," in *British Machine Vision Conference 2016*, 2016.

[13] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International Conference on Machine Learning*, pp. 6105–6114, PMLR, 2019.

[14] H. Huang, "Mechanisms of dimensionality reduction and decorrelation in deep neural networks," *Physical Review E*, vol. 98, no. 6, 2018.

[15] S. Recanatesi, M. Farrell, M. Advani, T. Moore, G. Lajoie, and E. Shea-Brown, "Dimensionality compression and expansion in deep neural networks," *arXiv preprint arXiv:1906.00443*, 2019.

[16] A. Ansuini, A. Laio, J. H. Macke, and D. Zoccolan, "Intrinsic dimension of data representations in deep neural networks," *Advances in Neural Information Processing Systems*, vol. 32, pp. 6111–6122, 2019.

[17] D. Bonet, "Improved neural network generalization using channel-wise nnk graph constructions," 2021.

[18] A. Elisseeff, M. Pontil, *et al.*, "Leave-one-out error and stability of learning algorithms with applications," *NATO science series sub series iii computer and systems sciences*, vol. 190, pp. 111–130, 2003.

[19] N. Aronszajn, "Theory of reproducing kernels," *Transactions of the American mathematical society*, vol. 68, no. 3, pp. 337–404, 1950.

[20] W. Dong, C. Moses, and K. Li, "Efficient k-nearest neighbor graph construction for generic similarity measures," in *Proceedings of the 20th international conference on World wide web*, pp. 577–586, 2011.

[21] V. Chvátal and P. Hammer, "Aggregations of inequalities," *Studies in Integer Programming, Annals of Discrete Mathematics*, vol. 1, pp. 145–162, 1977.

[22] J. Tropp and A. Gilbert, "Signal recovery from random measurements via orthogonal matching pursuit," *IEEE Trans. on info. theory*, 2007.

[23] L. Prechelt, "Early stopping-but when?," in *Neural Networks: Tricks of the trade*, pp. 55–69, Springer, 1998.

[24] Y. Yao, L. Rosasco, and A. Caponnetto, "On early stopping in gradient descent learning," *Constructive Approximation*, vol. 26, no. 2, pp. 289–315, 2007.

[25] G. Raskutti, M. J. Wainwright, and B. Yu, "Early stopping and non-parametric regression: an optimal data-dependent stopping rule," *The Journal of Machine Learning Research*, vol. 15, no. 1, 2014.

[26] M. Mahsereci, L. Balles, C. Lassner, and P. Hennig, "Early stopping without a validation set," *arXiv preprint arXiv:1703.09580*, 2017.

[27] D. Duvenaud, D. Maclaurin, and R. Adams, "Early stopping as non-parametric variational inference," in *Artificial Intelligence and Statistics*, pp. 1070–1077, PMLR, 2016.

[28] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," 2009.

[29] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic gradient descent," in *ICLR: International Conference on Learning Representations*, pp. 1–15, 2015.

[30] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[31] J. K. Terry, M. Jayakumar, and K. De Alwis, "Statistically significant stopping of neural network training," *arXiv preprint arXiv:2103.01205*, 2021.