# Growing Neural Networks have Flat Optima and Generalize Better

Paul Caillon, Christophe Cerisara

# Growing Neural Networks have Flat Optima and Generalize Better

Paul Caillon and Christophe Cerisara
Université de Lorraine, CNRS, LORIA
54500 Vandoeuvre-les-Nancy, France

September 13, 2024

**Abstract**

In this work, we study the loss landscape of growing neural networks and show that they have flatter minima than when trained with all of their parameters from random initialization. Then, we further evaluate and compare the generalization properties of both growing and non-growing models using, along with standard measures such as the training loss and the validation accuracy, an uncommon approximation of the population risk. The results we find suggest that growing models have better generalization properties. This supports the argument that flatness of the loss positively correlates with generalization in the current debate in the scientific community about flatness. We validate our approach on a wide range of binary Natural Language Processing tasks with large state-of-the-art deep learning models. Our theoretical and experimental results open new perspectives to study these questions through the prism of growing neural networks and risk approximations.

## 1 Introduction

Over the last few years, along with its practical successes in multiple research domains ; the theoretical properties of deep learning [25] have been a subject of active investigation, from the expressivity [26], [3] and the generalization properties to the trainability [9], [22] of a network.

Some empirical works observe that generalization and flatness of the minima found during training are related [7], [24]. Even though most Hessian-based flatness measures are sensible to re-scaling as pointed out in [13], some recent works [35] introduce a scale-invariant measure to show that smaller batch sizes correlate with flatter minima and better generalization performances. Sharper minima are thus believed to be sub-optimal and to be avoided during learning, despite the debate still being opened [46].

We study next these questions in a challenging application domain: Natural Language Processing (NLP), where models may range from millions to one trillion parameters and are typically fine-tuned on smaller task-specific corpora. Over-fitting and catastrophic forgetting during the fine-tuning process may thus severely impact their

generalization capabilities [20]. We argue that an alternative to standard fine-tuning might be to train a growing application-specific head on top of it. We thus study in this work whether adding neurons incrementally instead of learning them all from the beginning could achieve a better minimum for the neural network. In the following, we propose to intuitively justify and experimentally investigate this hypothesis by comparing an incrementally growing network with one with a fixed architecture learned from scratch, with competitive models.

## 2 Related Works

**Growing Neural Networks :** Standard Neural Networks typically use a given static architecture, with fixed number of layers and hidden units, which are designed thanks to time-consuming hand-engineering, through trial-and-error processes. Automating the architecture search is thus an active field of research, often called NAS (for Neural Architecture Search) [14], with proposals [11, 10] to use a grow-and-prune training paradigm to iteratively add and remove units, obtaining competitive neural networks that are much smaller then their static counterpart, the algorithms growing neural networks commonly answering one of the following questions : **where, when and how**. Growing Neural Networks (GrowNN) are often used in the field of continual learning [33] where the model must be able to learn new knowledge over long time spans without the new information interfering with previously learned knowledge (called catastrophic forgetting [31]). The possibility to transfer the knowledge from a previous smaller network to a new deeper or wider network is explored since [8], proposing an answer to "where to insert new neurons", both in existing layers or by creating new layers. In this field, recent works have shown that GrowNN starting from small networks, growing both wider and deeper can prevent catastrophic forgetting and achieve accurate and relatively small models compared to state-of-the-art models [28, 19, 43]. A recent work [16] also focus on finding the initialization of the inserted neurons, proposing an answer to "how to initialize the new neurons?". Those performances raise the question as to why GrowNN achieve these results, one idea being that as the parameters space topology becomes increasingly more complex through training, what was previously a local minima can become a saddle point and be escaped by growing, yielding to a monotonic decrease of the loss [41], in which authors require the training loss to attain plateau before adding new parameters, while other methods grow using a predefined schedule. Our work also contributes to this field of research that studies the loss landscape in the context of GrowNN, by proposing a new point-of-view, as to "why" growing neural networks might be commendable.

**Loss Landscape, Smooth Minima and Generalization :** The ability to generalize to unseen data is often linked to the flatness of the minima found during training as shown in empirical works [7, 24]: sharp minima lead to poorer generalization. The connections between loss landscapes, generalization performances and stochastic gradient descent (SGD) training have been important topics in machine learning for years. It has for example been shown that for large-size neural networks, the local minima of the loss function is often close to the global minima [9]. [44] show that SGD moves in regions of the loss landscape comparable to a valley. Most of the recent works on the global

properties of loss landscapes are discussed in [39] in which they review both theoretical and empirical findings. Sharpness-based analysis is an important sub-field of the current research on loss landscapes. A pivotal work [24] empirically shows that large-batch training leads to sharp local minima, making generalization worse and has recently been extended by [35] with a scale-invariant measure to escape the re-scaling problem mentioned in [13]. It has also been shown in [21] that sharpness-based metrics perform well when compared to other complexity metrics used to evaluate generalization. Using sharpness-based ideas, some works show that smoothing minima lead to an improvement of generalization performances. A notable work in this way, [17] recently proposed a procedure which seeks to both minimize the empirical loss and its sharpness, presenting empirical results with model generalization improvement across a wide variety of benchmark datasets.

Finally, as generalization is by essence the capacity of the model to be correct on every possible data points, the common train/validation/test paradigm is sometimes pointed out as biased as it is very difficult to construct a large-scale well-annotated dataset. To counter this issue, Deep Transfer Learning [40] uses test data from another dataset than the training data. For binary classification tasks a recent paper [6] proposes an approximation of the population risk that is theoretically less sensitive to over-fitting than the standard empirical risk approximation: we propose to use it next as a new indicator of generalization along with the standard validation accuracy to bring complementary evidence.

# 3 Growing neural networks and flatness

The relation between flatness of the loss landscape and generalization of the model is still subject to debates in the litterature. Although we do not claim to give a definitive answer to this question, we nevertheless bring new experimental evidence in Section 5 that supports this assumed relationship. In this Section 3, we first justify theoretically that, in the generic context of feed-forward neural network (FFNN), growing neural networks have in expectation a more flat minimum than standard networks.

Intuitively, the proof detailed next contrasts two training regimes:

- when considering the full search space with all parameters, the SGD converges towards an optimum that mainly depends on its distance to the initial parameters, and not on its flatness.
- when freezing one parameter (before growing), SGD is constrained to stay on an hyperplane H of the full search space; it then converges towards either a full-space optimum that intersects H, or a full-space saddle point that is a minimum on H: we may ignore for now these saddle points, as they do not change the conclusions of the proof as shown next.

The crux of the proof is that there are, in expectation, more flat minima on H than in the full space (proportionally), because flat minima occupy a larger volume than sharp ones in the full space, so the hyperplane H is more likely to intersect flat rather than sharp minima. Let us now detail this proof.

## 3.1 Justification

Let $R_e(\theta) \in \mathbb{R}^+$ be the empirical risk to minimize, with $\theta \in \mathbb{R}^d$ the model parameters. Let us assume that $R_e(\theta)$ has $N$ local minima: $(\theta_1^*, \dots, \theta_N^*)$ that are drawn uniformely in the domain of $\theta$. Each such local minimum is associated with a local volume $V(\theta_i^*)$, called the **basin of attraction**, which is defined as the volume around $\theta_i^*$ so that when a properly tuned SGD algorithm enters, it then almost surely converges towards $\theta_i^*$ [15]

In this subsection, we present a theoretical result that growing neural networks converge towards flatter minima than same-sized standard FFNN. To do so, we will compare next the expected volumes $V(\hat{\theta})$ of the local minima to which the SGD algorithm converges, respectively with and without a growing step. To do so we make the following hypothesis, that are further discussed in the Supplementary Material.

**- Hyp1:** Starting from random initialization parameters $\theta^0 \in \mathbb{R}^d$ that is not already within any basin of attraction, we assume that the SGD algorithm converges towards the closest local minimum $\hat{\theta} = \arg\min_i ||\theta_i^* - \theta^0||^2$. Note that this assumption can be replaced by a weaker version as discussed in the Supplementary Material, but this hypothesis makes the reasoning much easier to follow.

**- Hyp2:** $\theta^0$ does not start in any basin of attraction.

Let us start with the classical SGD that operates on the full $d$-dimensional domain. Because both $\theta^0$ and all $\theta_i^*$ are uniformly sampled in this domain, and because Hyp1 only depends on the distance between $\theta^0$ and the $\theta_i^*$, the SGD algorithm will randomly converge towards any of the minima, and

$$E[V(\hat{\theta})] = \sum_i p(\hat{\theta} = \theta_i^*) V(\theta_i^*) = \frac{1}{N} \sum_i V(\theta_i^*) \tag{1}$$

Let us now study this landscape for the growing model. It is easier to start from the final (post-growing) $\mathbb{R}^d$ space, and extract from it the $\mathbb{R}^{d-1}$ space that existed before the model grew: we assume next for clarity that we make the model grow by adding only a single parameter. Let us note $(O, \vartheta_1, \dots, \vartheta_d)$ the Cartesian coordinate system of the full $\mathbb{R}^d$ space. The $\mathbb{R}^d$ space is exactly the same as in the classical case with the same $\mathbb{R}^d$-dimensional local optima $(\theta_i^*)$. Starting from this $d$-dimensional space, we can "remove" one parameter dimension, say the first dimension $\vartheta_1$, to visualize the loss landscape before growing. "Removing" $\vartheta_1$ is equivalent to setting the first coordinate of any point to zero: $\theta_{i,1}^* = 0$, which defines an hyperplane $H$ on which training is performed before the growing phase. To compute the probability that $H$ intersects the volume around any $\mathbb{R}^d$-dimensional local optima $(\theta_i^*)$, we approximate this volume by the largest axis-aligned bounding box inscribed in the basin of attraction, which coordinates are:

$$[\theta_{i,1}^* - \tau_{i,1} : \theta_{i,1}^* + \tau'_{i,1}] \times \cdots \times [\theta_{i,d}^* - \tau_{i,d} : \theta_{i,d}^* + \tau'_{i,d}].$$

We can then relate the bounding box size to its volume : $V(\theta_i^*) \simeq \prod_{j=1}^d (\tau'_{i,j} - \tau_{i,j})$. We further assume a last hypothesis :

**- Hyp3:** $\tau'_{i,1} - \tau_{i,1}$ is conditionally independent with $(\tau'_{i,j} - \tau_{i,j})_{2 \leq j \leq d}$ given $V(\theta_i^*)$.

Hyp3 is motivated by the fact that there is no privileged direction for the basins of attraction; on the average, the width of the basin of attraction along the particular first

dimension is the same as along any other dimension, so:

$$E[V(\theta_i^*)] \simeq \prod_{j=1}^{d} E[\tau'_{i,j} - \tau_{i,j}] \simeq E[\tau'_{i,1} - \tau_{i,1}]^d \tag{2}$$

Let $p(H_i)$ be the probability that $H$ intersects the basin of attraction $V(\theta_i^*)$. Because the locations of the optima in the $\mathbb{R}^d$-dimensional space are uniformely distributed, and because $H$ follows the first dimension $\vartheta_1 = 0$, the probability that $H$ intersects $V(\theta_i^*)$ is proportional to $\tau'_{i,1} - \tau_{i,1}$. So, with $V_1$ the width of the domain along the first dimension,

$$p(H_i) = \frac{1}{V_1}(\tau'_{i,1} - \tau_{i,1}) \tag{3}$$

We have just derived two equations that intuitively state that $H$ is more likely to intersect a larger basin of attraction than a sharp one. We will use these equations later, but we first need to decompose the expectation according to the intermediary $\mathbb{R}^{d-1}$ optimum that appears in the growing process. Indeed, optimisation in the growing network proceeds as follows: first, $\theta^0$ is uniformely sampled on $H$. Then SGD converges towards $\hat{\theta}^H \in H$ that is within the basin of attraction of either one of the $\theta_i^*$, or of one of the $S$ saddle points $\{\theta_1^S, \dots, \theta_S^S\}$ on $H$. After that, one parameter is added to the model. And finally, SGD proceeds further in the full $\mathbb{R}^d$ space, starting from $\hat{\theta}^H$ and converging towards $\hat{\theta} \in \{\theta_1^*, \dots, \theta_N^*\}$. So the final average volume is $E[V(\hat{\theta})] = \sum_{i=1}^{N} V(\theta_i^*) p(\hat{\theta} = \theta_i^*)$.

The proof of the following Theorem 2 can be found in the Supplementary Material.

**Theorem 1.**

$$p(\hat{\theta} = \theta_i^*) = \frac{S + Np(H_i)}{N(S + \sum_j p(H_j))},$$

Theorem 2 gives : $E[V(\hat{\theta})] = \frac{1}{N} \sum_{i=1}^{N} V(\theta_i^*) \frac{S+Np(H_i)}{S+\sum_j p(H_j)}$.

So for growing models, the expected volume of the minima is a weighted average of all existing minima. It is easy to analyze which of these minima get a larger weight:

$$\frac{S + Np(H_i)}{S + \sum_j p(H_j)} > 1 \iff p(H_i) > E[p(H_i)].$$

From Eq-3 and Eq-2, we have $E[p(H_i)] = \frac{1}{V_1} E[V(\theta_i^*)]^{\frac{1}{d}}$ so we have actually partitioned the set of optima $\theta_i^*$ into two disjoint subsets:
- the largest optima with $p(H_i) > \frac{1}{V_1} E[V(\theta_i^*)]^{\frac{1}{d}}$
- the smallest optima with $p(H_i) \leq \frac{1}{V_1} E[V(\theta_i^*)]^{\frac{1}{d}}$.

Applying Eq- 2 to each subset, we can show that, on the average, the first subset corresponds to optima with a volume that is larger than the mean of all volumes. So, on the average, the minima that get a larger weight are also the minima with the largest volume. So the final expected volume is larger with growing models than with non-growing models that follow Eq- 1.

In order to make the final link between the volume and flatness, we may now further assume that a scale-independent flatness measure, such as proposed in [35], is correlated to the volume $V(\theta_i^*)$. Note that this correlation is subject to debates, see

for instance [46], however the authors note that flatness measures on average correlate with the volume. Assuming this result, our proof shows that growing neural networks converge in expectation towards optima that are flatter than classical neural networks.

### 3.2 Experimental Framework Presentation

Recent works [29, 23] propose to insert specific neurons in order to avoid bad minima. In the following, we investigate whether progressively inserting new neurons throughout training has an impact on the loss surface and the generalization performances. More precisely, our work's main focus is to explore whether growing approaches may intrinsically lead to topologically different minima in terms of flatness and generalization that may explain why growing approaches lead to better results than the standard ones in recent works. We do not explore advanced heuristics to grow the model, but adopt a simple growing strategy as we are interested in studying the properties of growing algorithms. In this way, we try not to answer one of the "where, when or how" questions about growing neural networks, but rather give insights on the "why". Furthermore, it has been shown in previous works [27, 32] that advanced NAS techniques are often only marginally superior to simple random search. We thus propose an experimental framework in which we answer the standard questions about growing as follows.
**When** : Neurons are incrementally inserted at a regular pace (with a constant time interval) during the learning phase. **Where** : The growth process can be seen as a standard insertion of a new layer, as done for example historically in [8], [38] or [36], achieved through multiple steps that are neuron insertions. **How** : The new neurons initialization follows the standard initialization scheme.

The goal here is to obtain a predefined target architecture at the end of the growing process, we thus have to insert a predefined number of neurons for each new layer to be able to compare strictly equivalent neural network architectures. Note that the proof in Section 3.1 does not rely on any specific algorithm, however we provide the exact insertion scheme and used algorithm in the Supplementary Material.

## 4 Evaluating generalization in neural models

In this section, we introduce another generalization metric that is complementary to the standard validation accuracy. Indeed, there are several potential issues with the validation accuracy: the validation corpus is often too similar from the training corpus to represent real-world and long-term generalization gap; it may also be too small, biased and have erroneous labels; it may also be too costly to annotate such a corpus for some applications. Finally, it is best practice to analyze multiple metrics instead of a single one to assess a model's property.

Most classifiers are trained by minimizing the empirical risk, which approximates the population risk on a finite training labeled corpus, but may also make the model overfit this corpus. So this empirical approximation might not be the best option to evaluate the generalization property of the trained model. Therefore, the standard approach rather computes the empirical risk on a held-out validation corpus. But this validation corpus might also not represent correctly the full population $p(x, y)$. We thus propose next to

exploit another approximation of the population risk, initially proposed in [1], that does not require any supervised label. This risk is based on the assumption that the class prior $P(y)$ is known and that the distribution of the classifier output scores may be well approximated by two real Gaussians with parameters $(\mu_0, \mu_1, \sigma_0, \sigma_1)$. The generalized central limit theorem justifies this assumption in high-dimensional space. The authors of [6] further show that, for binary classifiers and the hinge loss, this approximation of the risk leads to the following closed-form expression:

$$\begin{aligned} R(\mu,\sigma) = &\frac{P(y=0)}{2}(1+\mu_0)\left(1-\text{erf}\left(\frac{-1-\mu_0}{\sigma_0\sqrt{2}}\right)\right) + P(y=0)\sigma_0^2 N(-1;\mu_0,\sigma_0) \\ &+ \frac{P(y=1)}{2}(1-\mu_1)\left(1+\text{erf}\left(\frac{1-\mu_1}{\sigma_1\sqrt{2}}\right)\right) + P(y=1)\sigma_1^2 N(1;\mu_1,\sigma_1) \end{aligned} \tag{4}$$

where $N(x;\mu,\sigma)$ is the standard Normal probability density function. They further identify experimental conditions that satisfy the assumptions required in [1]. These conditions imply to compute the risk in a neighborhood of the empirical optimum, which is fulfilled in our experiments. The four Gaussian parameters are computed with any Gaussian Mixture Model estimation method applied to the set of model scores obtained after a forward pass over an (unlabeled) corpus.

We can note in Eq-4 that the risk does not directly depend on the training data, but it only depends on the four Gaussian parameters, which are themselves estimated on the whole training corpus, or on a large-enough batch. In other words, the Gaussian mixture may be viewed as a distribution that smooths the model output scores with only four learnable parameters. Consequently, $R(\mu,\sigma)$ only weakly depends on individual samples from the training dataset, and may thus be considered as quite robust to overfitting. In addition, this method is unsupervised, and $R(\mu,\sigma)$ may thus be estimated on larger collection of available data than labeled training sets. In our experiments, we have only computed the risk on the standard training corpus, but in a wide variety of applications, it is relatively easy to collect unlabeled data to further improve the estimation of the risk. Because this risk approximates the true classifier risk and is less sensitive to overfitting than the standard empirical risk, we propose to use it as an indicator of the generalization capabilities of our models and verify this claim experimentally.

# 5 Experimental Setup

The Metaeval [37] benchmark contains a collection of 101 NLP tasks for meta-learning and extreme multitask learning, including more than 50 binary classification tasks with very variable sizes, between a few hundred examples to hundreds of thousands examples.

## 5.1 Unsupervised Risk as a Generalization Measure

We study next the relation between the unsupervised risk described in Section 4 and the validation accuracy to assess the generalization capabilities of binary models. We propose to fine-tune RoBERTa [30] and its distilled counterpart DistilRoBERTa on

| Epoch | Test Accuracy | Training Corpus Metrics | |
|---|---|---|---|
| | | Emp. Risk | Unsup. Risk |
| Epoch 1 | 42.75% | 0.789 | 0.715058 |
| Epoch 2 | 61.08% | 0.536 | 0.592852 |
| Epoch 3 | 63.31% | 0.381 | 0.537434 |
| Epoch 4 | 64.58% | 0.252 | 0.355821 |
| Epoch 5 | 65.83% | 0.181 | 0.341291 |
| Epoch 6 | 66.04% | 0.119 | 0.340728 |
| Epoch 7 | 66.53% | 0.091 | 0.338699 |
| Epoch 8 | **67.27%** | 0.064 | **0.334629** |
| Epoch 9 | 65.59% | 0.047 | 0.341341 |
| Epoch 10 | 65.03% | **0.037** | 0.343245 |

Table 1: Evaluation of RoBERTa on CoLA. The empirical and unsupervised risks are computed on the training corpus. The accuracy is computed on the test corpus and measures generalization.

the binary classification tasks presented in Section 5 and to evaluate generalization performances and unsupervised risk.

BERT [12] is a well-known and reference contextual word embeddings model that is pre-trained with a Denoising Autoencoding objective and is at the basis of most state of the art results in many Natural Language Processing (NLP) tasks. RoBERTa builds on BERT's language masking strategy, but fine-tunes the original BERT model with a different choice of tasks and conditions. As most of today's state of the art NLP models, RoBERTa is a complex and large neural network, which thus faces issues related to over-parametrization. It is composed of 355 million parameters stored in 24 layers of self-attention, with a classification head on top to output the desired number of classes.

The results are reported in Table 3. Similar results on other datasets are available in the Additional Contents. We can note that the unsupervised risk is correlated to the test accuracy, which is the standard metric to measure generalization performances: when the test accuracy increases, the unsupervised risk decreases and vice-versa. The empirical and unsupervised risks measure different risks as the empirical risk decreases all the way through the 10 epochs, while the unsupervised risk attains its minimum at the 8th epoch and then increases. This minimum of the unsupervised risk corresponds to the best test accuracy, just before the accuracy decreases because of over-fitting. On these experiment, the unsupervised risk seems to be empirically less sensitive to over-fitting than the standard empirical risk approximation, which confirms the theoretical findings discussed in Section 4.

## 5.2 Growth of RoBERTA's classification head

The goal of this section is to check the correctness of the intuition formalized Section 3.1, namely whether the minimum found by a growing neural network is flatter than a minimum found by a standard FFNN. To do so, we propose to compare standard Transformers models for NLP classification tasks with their counterparts in which

we replaced the standard classification head with a growing classification head which ultimately will have the same architecture as the standard one. The classification head of RoBERTa (respectively DistilRoBERTa) is classically composed of a two-layers feed-forward network with 1024 (resp. 768) hidden neurons. We thus propose next to replace this classification head with a bare one-layer feed-forward network, and to grow it until it reaches the same number of parameters as the reference classification layer. The main pre-trained model, i.e., every parameter except the final classification head, is fine-tuned as such, without any further modifications.

The growth training process is studied and compared with a standard training process. We call "last", "valid" and "risk" the final models obtained by optimizing respectively the empirical loss, validation accuracy and unsupervised risk. Starting from the same model without hidden layer in the classification head, we proceed as follows, we train the model for three epochs, then add one third of the hidden parameters to the model state corresponding to the best epoch with respect to the chosen metric. We repeat this process until having full models (for a total of 12 epochs for each model). The experiments were run 5 times with random seed for each model. Some results for DistilRoBERTa on corpora from the Metaeval benchmark are presented in Table 5. Those corpora were chosen because of the variety of sizes, priors on the train set and relative difficulty. For a more detailed presentation of each corpus, the set of used hyper-parameters and other experimental results, see the Additional Contents. The goal here is not to achieve state-of-the-art results on the given corpora through extensive hyper-parameters search, but to compare standard and growing models with a given and coherent set of hyper-parameters in order to isolate the effect of the growth on the models' performances and loss surfaces.

| Dataset | Model | Emp. Risk | Val. Acc. | Spectral Norm | U. Risk |
|---|---|---|---|---|---|
| **Pers.-Eloquence** | Normal | **0.14 ± 0.02** | 0.7555±0.005 | $(1.60 \pm 0.58) \times 10^9$ | 0.45±0.02 |
| | Grow Last | 0.25 ± 0.03 | 0.755 ± 0.005 | $(3.3 \pm 0.2) \times 10^8$ | 0.46 ± 0.02 |
| | Grow Valid | 0.24 ± 0.04 | 0.760 ± 0.005 | $(1.5 \pm 0.4) \times 10^8$ | 0.45 ± 0.01 |
| | Grow Risk | 0.15 ± 0.01 | **0.767 ± 0.001** | $\mathbf{(1.2 \pm 0.3) \times 10^7}$ | **0.31 ± 0.03** |
| **Pers.-Specificity** | Normal | **0.24 ± 0.01** | 0.814 ± 0.006 | $(1.88 \pm 0.14) \times 10^9$ | 0.23 ± 0.05 |
| | Grow Last | 0.340 ± 0.007 | **0.823 ± 0.003** | $(2.15 \pm 0.07) \times 10^7$ | **0.12 ± 0.04** |
| | Grow Valid | 0.26 ± 0.02 | 0.822 ± 0.004 | $\mathbf{(1.14 \pm 0.52) \times 10^7}$ | 0.14 ± 0.01 |
| | Grow Risk | 0.360 ± 0.004 | 0.816 ± 0.002 | $(3.64 \pm 0.22) \times 10^7$ | 0.21 ± 0.07 |
| **CoLA** | Normal | **0.11 ± 0.02** | 0.57 ± 0.02 | $(4.61 \pm 0.05) \times 10^9$ | 0.41 ± 0.02 |
| | Grow Last | 0.15 ± 0.01 | 0.57 ± 0.02 | $(2.35 \pm 0.03) \times 10^9$ | 0.39 ± 0.04 |
| | Grow Valid | 0.120 ± 0.005 | **0.60 ± 0.01** | $\mathbf{(2.03 \pm 0.09) \times 10^8}$ | **0.317 ± 0.012** |
| | Grow Risk | 0.13 ± 0.02 | 0.594 ± 0.016 | $(7.65 \pm 0.31) \times 10^8$ | 0.33 ± 0.02 |
| **Emobank-Dom.** | Normal | **0.29 ± 0.02** | 0.63 ± 0.01 | $(9.8 \pm 0.5) \times 10^8$ | 0.29 ± 0.04 |
| | Grow Last | 0.53 ± 0.04 | 0.62 ± 0.02 | $(1.07 \pm 0.30) \times 10^9$ | 0.38 ± 0.03 |
| | Grow Valid | 0.35 ± 0.01 | **0.69 ± 0.02** | $\mathbf{(6.22 \pm 0.18) \times 10^8}$ | **0.21 ± 0.04** |
| | Grow Risk | 0.412 ± 0.015 | 0.64 ± 0.01 | $(6.7 \pm 0.3) \times 10^8$ | 0.27 ± 0.02 |
| **Justice** | Normal | **0.23 ± 0.01** | **0.57 ± 0.01** | $(2.1 \pm 0.5) \times 10^9$ | 0.16 ± 0.05 |
| | Grow Last | 0.35 ± 0.01 | 0.55 ± 0.01 | $\mathbf{(1.22 \pm 0.14) \times 10^9}$ | 0.12 ± 0.01 |
| | Grow Valid | 0.35 ± 0.07 | 0.56 ± 0.03 | $(1.36 \pm 0.12) \times 10^9$ | 0.11 ± 0.02 |

| Dataset | Model | Emp. Risk | Val. Acc. | Spectral Norm | U. Risk |
|---|---|---|---|---|---|
| | Grow Risk | $0.36 \pm 0.04$ | $0.56 \pm 0.02$ | $(1.78 \pm 0.31) \times 10^9$ | $\mathbf{0.10 \pm 0.03}$ |
| **Offensive** | Normal | $\mathbf{0.06 \pm 0.01}$ | $0.79 \pm 0.01$ | $(3.02 \pm 0.31) \times 10^9$ | $0.420 \pm 0.013$ |
| | Grow Last | $0.19 \pm 0.02$ | $0.80 \pm 0.01$ | $(4.78 \pm 0.17) \times 10^8$ | $0.37 \pm 0.02$ |
| | Grow Valid | $0.11 \pm 0.01$ | $\mathbf{0.845 \pm 0.014}$ | $\mathbf{(8.50 \pm 0.09) \times 10^7}$ | $\mathbf{0.22 \pm 0.01}$ |
| | Grow Risk | $0.17 \pm 0.03$ | $0.84 \pm 0.02$ | $(9.3 \pm 0.2) \times 10^7$ | $0.25 \pm 0.02$ |
| **Virtue** | Normal | $\mathbf{0.085 \pm 0.005}$ | $0.68 \pm 0.01$ | $(2.40 \pm 0.15) \times 10^9$ | $1.04 \pm 0.02$ |
| | Grow Last | $0.27 \pm 0.01$ | $0.73 \pm 0.02$ | $(2.40 \pm 0.34) \times 10^8$ | $0.57 \pm 0.03$ |
| | Grow Valid | $0.30 \pm 0.02$ | $\mathbf{0.80 \pm 0.03}$ | $\mathbf{(3.50 \pm 0.37) \times 10^7}$ | $\mathbf{0.34 \pm 0.02}$ |
| | Grow Risk | $0.23 \pm 0.04$ | $0.76 \pm 0.02$ | $(9.30 \pm 0.11) \times 10^8$ | $0.40 \pm 0.05$ |

Table 2: Evaluation of the proposed growing network on corpora from the Metaeval benchmark with DistilRoBERTa base. The empirical risk, unsupervised risk and spectral norm are computed on the training corpus.

# 6 Discussion

Table 3 suggests that the unsupervised risk described in Section 4 might be used as a proxy to assess the generalization capabilities of binary models when a corpus representative of unknown or future test conditions is not available. We then use this unsupervised risk as a growth criterion and present our results in Table 5. The results presented in Table 5 suggest that growing neural networks indeed achieve a flatter minimum than their standard counter parts and that flatter minima tend to generalize better, no matter the generalization metric used. In fact, apart from the Justice data set where the results on every metric are really close to each other no matter the model, we can observe that the empirical and unsupervised risks indeed measure different risks: the empirical risk is always better for the standard model than for the growing models, which intuitively results from the fact that the standard models have more time to update all of their parameters simultaneously, and thus reach better minima. Conversely, most growing models have a better unsupervised risk. It may be possible to view growing a network as a form of regularization, but note that the reference models are also trained with all best practices in terms of regularization strategies, as designed in [30]. The lowest unsupervised risk reached by any model also often correlates with the flatter loss landscape, as measured by the spectral norm. Lastly, with our simple growth paradigm, choosing to grow the best model with respect to a chosen metric (whether the validation accuracy or the unsupervised risk) seems to lead in general to better results than simply growing the last model. The best metric to choose for growing seems to be the validation accuracy, which can be explained by the fact that validation accuracy is also our final evaluation criterion, the "growing valid" models thus acting as oracles with regard to this metric. In practice, the good results obtained when growing from the unsupervised risk are more interesting as no validation corpus is required.

The proposed approach has two main fundamental limitations. First, the growing process introduces an additional hyper-parameter as compared to the standard training

process, that is the frequency at which to grow the model. This hyper-parameter shall be tuned in addition to the standard hyper-parameters to tune, which may increase the development time. Hopefully, this hyper-parameter may not be the most difficult to tune, as we have observed that the intuitive equi-repartition of the growing steps within standard number of epochs give reasonably good results, at least in our experiments. Second, the precise growing procedure shall be adapted to every type of neural network and training procedure; for instance, recent neural architectures may exploit sparse mixture of experts instead of classical feed-forward networks, or advanced NAS algorithms may be used to design the model. In such cases, the growing process has to be adapted to the specifics of the system, for instance to decide which parts of the network shall be grown (the gating model and/or each expert vs. adding a new expert?) and to ensure a smooth interaction between the growing process and the NAS algorithm. This gives flexibility and control means to the developer, at the cost of an increased complexity and design time.

# 7 Conclusion

In this work, we give both theoretical and experimental evidence that growing neural networks converge in expectation towards flatter minima than a comparable network initialized with all parameters. We further show experimentally that these flatter minima better generalize than the standard training regime when growing the classification head of state-of-the-art large NLP neural networks. We study this generalization property of models both with a standard metric, validation accuracy, and with an uncommon approximation of the population risk that does not require annotated labels, and may thus advantageously complement the validation accuracy and even replace it when the validation corpus is missing or when future test conditions are hard to predict.

In summary, we established new connections between growing neural networks, flatness of the loss surface and generalization of neural networks. These connections give a fresh view on the study of properties of deep neural networks, and open novel perspectives to explore in the three field respectively concerned by generalization, progressive networks and classifier risk approximation. In future works, we also plan to investigate more concrete applications of this work, such as model selection and early stopping when there is no available validation data to approximate the mismatch between training and unknown test conditions.

# References


[1] K. Balasubramanian, P. Donmez, and G. Lebanon. Unsupervised supervised learning II: Margin-based classification without labels. *JMLR*, 12:3119–3145, 2011.

[2] Francesco Barbieri, Jose Camacho-Collados, Leonardo Neves, and Luis Espinosa-Anke. TweetEval: Unified Benchmark and Comparative Evaluation for Tweet Classification. *arXiv e-prints*, page arXiv:2010.12421, October 2020.

[3] A. R Barron. Approximation and estimation bounds for artificial neural networks. *Machine learning*, 14(1):115–133, 1994.

[4] Sven Buechel and Udo Hahn. EmoBank: Studying the impact of annotation perspective and representation format on dimensional emotion analysis. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 578–585, Valencia, Spain, April 2017. Association for Computational Linguistics.

[5] Winston Carlile, Nishant Gurrapadi, Zixuan Ke, and Vincent Ng. Give me more feedback: Annotating argument persuasiveness and related attributes in student essays. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 621–631, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[6] C. Cerisara, P. Caillon, and G. Le Berre. Unsupervised post-tuning of deep neural networks. In *IJCNN*, pages 1–8. IEEE, 2021.

[7] P. Chaudhari, A. Choromanska, S. Soatto, Y. LeCun, C. Baldassi, J. Borgs, C.and Chayes, L. Sagun, and R. Zecchina. Entropy-sgd: Biasing gradient descent into wide valleys. *Journal of Statistical Mechanics: Theory and Experiment*, 2019(12):124018, 2019.

[8] Tianqi Chen, Ian Goodfellow, and Jonathon Shlens. Net2Net: Accelerating Learning via Knowledge Transfer. *arXiv e-prints*, page arXiv:1511.05641, November 2015.

[9] A. Choromanska, M. Henaff, M. Mathieu, G. Ben Arous, and Y. LeCun. The loss surfaces of multilayer networks. In *Artificial intelligence and statistics*, pages 192–204. PMLR, 2015.

[10] X. Dai, H. Yin, and N. K Jha. Grow and prune compact, fast, and accurate lstms. *IEEE Transactions on Computers*, 69(3):441–452, 2019.

[11] X. Dai, H. Yin, and N. K Jha. Nest: A neural network synthesis tool based on a grow-and-prune paradigm. *IEEE Transactions on Computers*, 68(10):1487–1497, 2019.

[12] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT (1)*, 2019.

[13] L. Dinh, R. Pascanu, S. Bengio, and Y. Bengio. Sharp minima can generalize for deep nets. In *ICML*, pages 1019–1028. PMLR, 2017.

[14] T. Elsken, J. H. Metzen, and F. Hutter. Neural architecture search: A survey. *The Journal of Machine Learning Research*, 20(1):1997–2017, 2019.

[15] Dumitru Erhan, Aaron Courville, Yoshua Bengio, and Pascal Vincent. Why does unsupervised pre-training help deep learning? In *AISTATS*, 2010.

[16] Utku Evci, Bart van Merrienboer, Thomas Unterthiner, Fabian Pedregosa, and Max Vladymyrov. Gradmax: Growing neural networks using gradient information. In *ICLR*, 2022.

[17] P. Foret, A. Kleiner, H. Mobahi, and B. Neyshabur. Sharpness-aware minimization for efficiently improving generalization. In *ICLR*, 2021.

[18] Dan Hendrycks, Collin Burns, Steven Basart, Andrew Critch, Jerry Li, Dawn Song, and Jacob Steinhardt. Aligning AI With Shared Human Values. *arXiv e-prints*, page arXiv:2008.02275, August 2020.

[19] C.-Y. Hung, C-H. Tu, C.-E. Wu, Y.-M. Chen, C.-H.and Chan, and C.-S. Chen. Compacting, picking and growing for unforgetting continual learning. In *NeurIPS*, volume 32, 2019.

[20] Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. SMART: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. In *ACL*, 2020.

[21] Y.* Jiang, B.* Neyshabur, H. Mobahi, D. Krishnan, and S. Bengio. Fantastic generalization measures and where to find them. In *ICLR*, 2020.

[22] K. Kawaguchi. Deep learning without poor local minima. In *NeurIPS*, pages 586–594, 2016.

[23] K. Kawaguchi and L. Kaelbling. Elimination of all bad local minima in deep learning. In *AISTATS*, pages 853–863. PMLR, 2020.

[24] Nitish Shirish Keskar, Jorge Nocedal, Ping Tak Peter Tang, Dheevatsa Mudigere, and Mikhail Smelyanskiy. On large-batch training for deep learning: Generalization gap and sharp minima. In *ICLR*, 2017.

[25] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature Cell Biology*, 521(7553), May 2015.

[26] M. Leshno, V. Ya Lin, A. Pinkus, and S. Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural networks*, 6, 1993.

[27] L. Li and A. Talwalkar. Random search and reproducibility for neural architecture search. In *UAI*, pages 367–377. PMLR, 2020.

[28] X. Li, Y. Zhou, T. Wu, R. Socher, and C. Xiong. Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting. In *ICML*, pages 3925–3934. PMLR, 2019.

[29] S. Liang, R. Sun, J. D Lee, and R. Srikant. Adding one neuron can eliminate all bad local minima. *NeurIPS*, 31:4350–4360, 2018.

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

[31] Michael McCloskey and Neal J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. Psychology of Learning and Motivation. Academic Press, 1989.

[32] R. Negrinho, M. Gormley, G. J Gordon, D. Patil, N. Le, and D. Ferreira. Towards modular and programmable architecture search. *NeurIPS*, 32, 2019.

[33] G. I. Parisi, R. Kemker, J. L. Part, C Kanan, and S. Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, 2019.

[34] Guruprasad Raghavan and Matt Thomson. Neural networks grown and self-organized by noise. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 1895–1905, 2019.

[35] A. Rangamani, N. H Nguyen, A. Kumar, D. Phan, S. H Chin, and T. D Tran. A scale invariant flatness measure for deep network minima. *arXiv preprint arXiv:1902.02434*, 2019.

[36] Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Jie Tan, Quoc Le, and Alex Kurakin. Large-Scale Evolution of Image Classifiers. *arXiv e-prints*, page arXiv:1703.01041, March 2017.

[37] Damien Sileo and Marie-Francine Moens. Analysis and Prediction of NLP Models Via Task Embeddings. *ArXiv*, abs/2112.05647, 2021.

[38] Masanori Suganuma, Shinichi Shirakawa, and Tomoharu Nagao. A Genetic Programming Approach to Designing Convolutional Neural Network Architectures. *arXiv e-prints*, page arXiv:1704.00764, April 2017.

[39] R. Sun, D. Li, S. Liang, T. Ding, and R. Srikant. The global landscape of neural networks: An overview. *IEEE Signal Processing Magazine*, 37(5):95–108, 2020.

[40] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu. A survey on deep transfer learning. In *ICANN*, pages 270–279. Springer, 2018.

[41] D Wang, M Li, L Wu, V Chandra, and Q Liu. Energy-aware neural architecture optimization with fast splitting steepest descent, 2020.

[42] A. Warstadt, A. Singh, and S. R Bowman. Neural network acceptability judgments. *Transactions of the ACL*, 7:625–641, 2019.

[43] L. Wu, B. Liu, P. Stone, and Q. Liu. Firefly neural architecture descent: a general approach for growing neural networks. In *NeurIPS*, 2020.

[44] C. Xing, D. Arpit, C. Tsirigotis, and Y. Bengio. A Walk with SGD. *arXiv e-prints*, 2018.

[45] Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. SemEval-2019 task 6: Identifying and categorizing offensive language in social media (OffensEval). In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 75–86, Minneapolis, Minnesota, USA, June 2019. Association for Computational Linguistics.

[46] Shuofeng Zhang, Isaac Reid, Guillermo Valle Pérez, and Ard Louis. Why flatness does and does not correlate with generalization for deep neural networks. *arXiv e-prints*, page arXiv:2103.06219, March 2021.

# A Appendix

## A.1 Detailed Analysis of the Link between Growing Neural Networks and Flatness

### A.1.1 Proof of Theorem 1

In this subsection we detail the proof of the Theorem 1 presented in the main paper. As a remainder, we defined :

- the random initialization parameters $\theta^0 \in \mathbb{R}^d$ and we assumed that the SGD algorithm converges towards the closest local minimum $\hat{\theta} = \arg\min_i ||\theta_i^* - \theta^0||^2$. Note that this assumption (**Hyp 1**) can be replaced by a weaker version as detailed next.
- the hyperplane H of the full search space. When freezing one parameter (before growing), SGD is constrained to stay on an hyperplane H of the full search space and then converges towards either a full-space optimum that intersects H, or a full-space saddle point that is a minimum on H.
- $p(H_i)$, the probability that $H$ intersects the basin of attraction $V(\theta_i^*)$.
- As we detailed in the main paper, optimisation in the growing network proceeds as follows:
  - First, $\theta^0$ is uniformely sampled on $H$.
  - Then SGD converges towards $\hat{\theta}^H \in H$ that is within the basin of attraction of either one of the $\theta_i^*$, or of one of the $S$ saddle points $\{\theta_1^S, \ldots, \theta_S^S\}$ on $H$.
  - After that, one parameter is added to the model.
  - Finally, SGD proceeds further in the full $\mathbb{R}^d$ space, starting from $\hat{\theta}^H$ and converging towards $\hat{\theta} \in \{\theta_1^*, \ldots, \theta_N^*\}$.

Then, we have the following results :

**Theorem 1.** $\sum_{j=1}^{S} p(\hat{\theta} = \theta_i^*, \hat{\theta}^H = \theta_j^S) = \frac{S}{N(S+K)}$.
*with $K = \sum_i p(H_i)$ the number of real optima whose basin of attraction intersects with $H$.*

*Proof.* $p(\hat{\theta} = \theta_i^*, \hat{\theta}^H = \theta_j^S) = p(\hat{\theta} = \theta_i^* | \hat{\theta}^H = \theta_j^S) p(\hat{\theta}^H = \theta_j^S)) = \frac{1}{N} \times \frac{1}{(S+K)}$. □

**Theorem 2.** $\sum_{j=1}^{N} p(\hat{\theta} = \theta_i^*, \hat{\theta}^H \sim \theta_j^*) = p(\hat{\theta}^H \sim \theta_i^*)$.
*where $\hat{\theta}^H \sim \theta_j^*$ denote that $\hat{\theta}^H$ has converged on $H$ towards a point within the basin of attraction of $\theta_j^*$, which is only possible when $H$ intersects $V(\theta_j^*)$.*

*Proof.* $p(\hat{\theta} = \theta_i^*, \hat{\theta}^H \sim \theta_j^*) = p(\hat{\theta} = \theta_i^* | \hat{\theta}^H \sim \theta_j^*) p(\hat{\theta}^H \sim \theta_j^*)$. By definition of a basin of attraction, $p(\hat{\theta} = \theta_i^* | \hat{\theta}^H \sim \theta_i^*) = 1$ and $p(\hat{\theta} = \theta_i^* | \hat{\theta}^H \sim \theta_{j \neq i}^*) = 0$. □

**Theorem 3.** $p(\hat{\theta}^H \sim \theta_i^*) = \frac{p(H_i)}{S+K}$

*Proof.* $p(\hat{\theta}^H \sim \theta_i^*) = p(\hat{\theta}^H \sim \theta_i^* | H_i) p(H_i) + p(\hat{\theta}^H \sim \theta_i^* | \bar{H}_i)(1 - p(H_i))$.
The second term is null when $H$ does not intersect with $V(\theta_i^*)$, so
$p(\hat{\theta}^H \sim \theta_i^*) = p(\hat{\theta}^H \sim \theta_i^* | H_i) p(H_i)$.
Because of the uniform distribution of the critical points, the probability that SGD converges first towards any of the critical points that are on $H$ is uniform, so
$p(\hat{\theta}^H \sim \theta_i^* | H_i) = \frac{1}{S+K}$ □

**Theorem 2.** $p(\hat{\theta} = \theta_i^*) = \frac{S + N p(H_i)}{N(S + \sum_j p(H_j))}$

*Proof.* $p(\hat{\theta} = \theta_i^*) = \sum_{j=1}^{N} p(\hat{\theta} = \theta_i^*, \hat{\theta}^H \sim \theta_j^*) + \sum_{j=1}^{S} p(\hat{\theta} = \theta_i^*, \hat{\theta}^H = \theta_j^S)$.
By Lemmas 1, 2, we have : $p(\hat{\theta} = \theta_i^*) = p(\hat{\theta}^H \sim \theta_i^*) + \frac{S}{N(S+K)}$.
Lemma 3 allows to conclude. □

### A.1.2 Discussion over the needed hypothesis

Let us now discuss the validity of the various assumptions made in the main paper regarding the link between GrowNN and flat minima.

Our assumptions were:

-Minima are drawn uniformly: the proof stays correct if we replace it by a weaker assumption: the volume of every minimum is independent of its location, which is intuitively reasonable

**- Hyp1 :** SGD converges towards the closest minimum: as explained in the paper, we can use a weaker version: SGD converges towards a minimum with a probability that is proportional to its distance, which is also reasonable. However, we assumed Hyp1 as it made the reasoning much easier to follow.

**- Hyp2 :** Initial parameters are not already at a minimum: even if they occur, such rare cases do not alter the main conclusion of the proof. This hypothesis is not strictly necessary, as the situation is quite straightforward when the SGD algorithm already starts within a basin of attraction and this does not affect the main conclusion, but Hyp2 greatly simplifies the following derivation of the expected volume by enabling us to omit detailing multiple rare edge cases.

**- Hyp3 :** No privileged direction for basins of attraction: constraining the growing algorithm according to such eventual privileged direction will keep our proof correct, at the cost of more complex growing algorithm
-The volume of basins of attraction is correlated with flatness: according to [46], flatness measures on average correlate with the volume.
- We also assumed that there exists around any minimum an axis-aligned hypercube, so that any well-calibrated SGD entering into it will converge with probability 1 to this optimum: it is the largest such hypercube circumscribed inside the basin of attraction. This approximation has only 2 effects: first we need to wait longer for SGD to converge into this hypercube when it enters the real basin of attraction, second our intersection probability p(H) is actually slightly larger than what we compute, but our conclusions stay the same.

## A.2 Experimental Details and Additional Experimental Results

In this section, we present the whole set of hyper-parameters used to achieve the experimental results we present. We furthermore present additional experimental results that we could not include in the main paper due to length constraints.

### A.2.1 Detailed Experimental Conditions

In this subsection, we present the exact set of hyper-parameters used to achieve the experimental results we present in the main paper and in the following subsections. The code for these experiments will be publicly released and open-sourced upon publication. The experiments were run on a single TITAN X GPU. The most time-consuming part of the experiment was to evaluate the flatness measure developed by [35] using their code available at https://github.com/akshay-r/scale-invariant-flatness, which took several hours of calculation for each data-set (the exact time for each data-set being depending on the size of each data-set).

5 runs with random seeds were done for each experimental results we present. The mean and variance of these 5 runs are usually presented (when no variance is presented, it is because it was very low when compared to the mean). For each experiments, the used hyper-parameters are the following :

- Initial Learning Rate is $2 \times 10^{-5}$
- The total number of epochs is 12 (3 epochs for each size for growing neural networks : 3 epochs with no hidden neurons, 3 epochs with $\frac{1}{3}$of all hidden parameters, 3 epochs with $\frac{2}{3}$ and 3 epochs with all hidden parameters)
- We evaluate all the metrics (loss, validation accuracy, unsupervised risk...) every epoch.
- The other hyper-parameters are set to their default value in the Hugging Face Transformers Training Arguments [1], with for example a batch size of 8.

[1] https://huggingface.co/docs/transformers/v4.19.2/en/main_classes/trainer

The datasets we used are the following :

- CoLA : The Corpus of Linguistic Acceptability (CoLA) is a set of 10,657 English sentences labeled as grammatical or ungrammatical from published linguistics literature. [42]
- Persuasion [5] : is a collection of arguments from student essays annotated with factor of persuasiveness (Specificity, Eloquence, Relevance and Strength) with respect to a claim.
- Emobank [4] aggregates emotion annotations on text from various domains using the VAD representation format. Valence is defined as *corresponding to the concept of polarity*, Arousal as *degree of calmness or excitement* and Dominance as *perceived degree of control over a situation*.
- The ETHICS dataset [18] has contextualized scenarios about justice, deontology, virtue, ethics, utilitarianism and commonsense moral intuitions.
- The Tweet Eval benchmark [2] is composed of several tasks and we more specifically study the offensive language identification which consists in identifying whether some form of offensive language is present in a tweet. It relies on the SemEval2019 OffensEval dataset [45]

#### A.2.2 Unsupervised Risk as a Generalization Measure : Additional Experimental Results

In this subsection, we present additional experimental results to further justify the use of the Unsupervised Risk proposed by [6] as a Generalization Measure. The following Tables 3, 4 are the evaluation of DistilRoBERTa on different datasets from the Metaeval benchmark [37]. The empirical and unsupervised risks (respectively E. Risk and U. Risk) are computed on the training corpora, while the accuracy is computed on the test corpora and measures generalization. Note that the goal here is not to achieve state of the art results, but rather to show that the unsupervised risk is indeed less sensible to over-fitting than the empirical risk and can be used as a generalization measure.

#### A.2.3 Growth of RoBERTa's classification head : RoBERTa large as backbone

In this subsection we present a few additional results obtained with the RoBERTa large architecture as backbone (instead of the DistilRoBERTa base architecture for the results presented in the main paper). Those results presented in Table 5 suggest that the claims made in the main part of our work still hold with other architectures.

| Dataset | Model | Emp. Risk | Val. Acc. | Spectral Norm | U. Risk |
|---|---|---|---|---|---|
| **Commonsense** | Normal | **0.25 ± 0.03** | $0.726 \pm 0.004$ | $(8.3 \pm 0.4) \times 10^9$ | $0.20 \pm 0.02$ |
| | Growing Last | $0.31 \pm 0.03$ | $0.735 \pm 0.004$ | $(5.3 \pm 0.3) \times 10^9$ | $0.17 \pm 0.02$ |
| | Growing Valid | $0.33 \pm 0.02$ | $0.73 \pm 0.01$ | $(1.5 \pm 0.4) \times 10^9$ | $0.18 \pm 0.04$ |

*Continued on next page*

| Dataset | Model | Emp. Risk | Val. Acc. | Spectral Norm | U. Risk |
|---|---|---|---|---|---|
| | Growing Risk | $0.32 \pm 0.01$ | **0.740 ± 0.023** | **(1.3 ± 0.4) × $10^9$** | **0.09 ± 0.01** |
| **CoLA** | Normal | **0.978 ± 0.002** | $0.854 \pm 0.005$ | $(1.7 \pm 0.5) \times 10^{10}$ | $0.33 \pm 0.02$ |
| | Grow Last | $1.359 \pm 0.003$ | $0.852 \pm 0.002$ | $(9.2 \pm 0.2) \times 10^9$ | $0.33 \pm 0.02$ |
| | Grow Valid | $1.329 \pm 0.004$ | $0.858 \pm 0.001$ | $(4.3 \pm 0.1) \times 10^9$ | $0.31 \pm 0.01$ |
| | Grow Risk | $1.107 \pm 0.001$ | **0.869 ± 0.001** | **(3.6 ± 0.3) × $10^9$** | **0.30 ± 0.01** |

Table 5: Evaluation of the proposed growing network on corpora from the Metaeval benchmark. The empirical risk, unsupervised risk and spectral norm are computed on the training corpus. The accuracy is computed on the validation corpus. The backbone architecture is RoBERTA large

## A.3 Algorithms and Exact Growth Settings

### A.3.1 Exact Growth Settings

The experiments were conducted with a constrained type of neuron insertion process :

- a three stages insertion process, meaning the model grows new neurons three times during training to achieve its final architecture.
- with the standard initialization scheme, meaning all the neurons follow a uniform distribution between -$\frac{1}{\sqrt{I}}$ and $\frac{1}{\sqrt{I}}$ to follow the initialization scheme of a Linear layer in Pytorch [2].

The inserted neurons are thus initialized in the same way as those in the standard feedforward neural networks we compare our network with. In this way, the insertion process is the only difference between the two training methods, every hyper-parameter being equal otherwise. This enables us to study several possible selection criteria to choose the model from which growing shall proceed.

### A.3.2 General Algorithm

In this section we present a general framework for growing neural networks.

Having a set of input and output vectors $(x, y) \in \mathbf{R}^{nxm}$, the minimal feed forward neural network is one with only an output layer, thus having $\forall i \in [1, m]$ :

$$y_i(x, W) = \sum_{j \in \Pi(i)} \sigma_j(x_j \cdot w_{j,i}).$$

The underlying idea in Algorithm 0 is to start with such a model and to increase the number of learnable parameters with the number of epochs, without any constraints on the final network obtained with this procedure. As mentioned in [34], a model growth can be divided in two types : a horizontal growth in which the size of a layer is increasing and a vertical growth in which the number of layers is increasing. Intuitively, our growing procedure can be seen as a naive insertion process :

Let's consider a random neural network of total depth $D$ : first, an existing neuron $H$ on depth $l, 0 \leq l < D$ is randomly chosen and a new neuron $K$, which inherits some its children, is created. $K$ also gets new parent connections with existing neurons randomly chosen in the previous layers and can get random children connection in the subsequent layers.

[2] see https://pytorch.org/docs/stable/generated/torch.nn.Linear.html

| Epoch | Test Acc. | E. Risk | U. Risk |
|---|---|---|---|
| Epoch 1 | 0.7209 | 0.3227 | 0.7914 |
| Epoch 2 | 0.7860 | 0.2593 | 0.5267 |
| Epoch 3 | 0.7883 | 0.2212 | 0.4853 |
| Epoch 4 | 0.8000 | 0.1984 | 0.4741 |
| Epoch 5 | **0.8000** | 0.1767 | 0.4443 |
| Epoch 6 | 0.7883 | 0.1485 | **0.3945** |
| Epoch 7 | 0.7837 | 0.1353 | 0.4142 |
| Epoch 8 | 0.7779 | 0.1145 | 0.4193 |
| Epoch 9 | 0.7651 | 0.0876 | 0.4039 |
| Epoch 10 | 0.7750 | **0.0789** | 0.4216 |

| Epoch | Test Acc. | E. Risk | U. Risk |
|---|---|---|---|
| Epoch 1 | 0.4778 | 0.6204 | 0.8484 |
| Epoch 2 | 0.4939 | 0.5969 | 0.7754 |
| Epoch 3 | 0.5109 | 0.5131 | 0.6021 |
| Epoch 4 | 0.5316 | 0.4524 | 0.4309 |
| Epoch 5 | 0.5645 | 0.4219 | 0.3154 |
| Epoch 6 | 0.5702 | 0.3973 | 0.2827 |
| Epoch 7 | **0.5929** | 0.3536 | **0.2440** |
| Epoch 8 | 0.5771 | 0.2867 | 0.2517 |
| Epoch 9 | 0.5825 | 0.2248 | 0.2481 |
| Epoch 10 | 0.5903 | **0.2242** | 0.2716 |

Table 3: Evaluation of DistilRoBERTa on Offensive (left) and CoLA(right)

Note that if $K$ inherits none of $H$'s children, the $l+1$'s layer size is increased by one as at insertion time none of $K$'s children can be on the $l+1$ layer and $K$ is added to this particular layer. On the contrary, if $K$ inherits one of $H$'s children that was on the $l+1$ layer at insertion time, the $l+2$ is populated by this particular neuron, meaning that if for example $l = D-1$, a new layer is created as the number of output units is fixed.

Let $\mathcal{P}_t$ be the set of all non-output neurons at insertion step $t$. Let $\mathcal{I}$ be the set of input neurons, of size I.

Initially, $\mathcal{P}_0 = \{1, ..., n_I\}$ only contains the $I$ input neurons, because we start without any hidden neuron. The weight matrix $W_0$ is upper triangular with non-null transitions from the input to output neurons that are initialized randomly, e.g., with Glorot or the uniform distribution $\mathcal{U}$ (the chosen initialisation scheme will be denoted $Init()$ in the following Algorithm 0). The depth of the input neurons is 0, and the depth of the output neurons is 1. Then, the growing step proceeds as shown in Algorithm 0.

As we consider the network as an unweighted directed acyclic graph, its topological ordering is easily derived from a Depth-First Search.

Let's suppose we use the General Insertion Algorithm to insert one neuron at a time, and let's explore what kind of neural networks can be achieved through repeated calls of this growth procedure. Each neuron inserted is inserted strictly between the input and output layers. Let's take a random network of random size with inputs of size $n$ and outputs of size $m$. This network can be represented by its Directed Acyclic Graph, and its weight matrix $W_N$ (with $N$ the number

| Epoch | Test Acc. | E. Risk | U. Risk |
|---|---|---|---|
| Epoch 1 | 0.5806 | 0.6807 | 0.7497 |
| Epoch 2 | 0.6935 | 0.4912 | 0.4680 |
| Epoch 3 | 0.7580 | 0.446 | 0.2442 |
| Epoch 4 | 0.7580 | 0.4182 | 0.1470 |
| Epoch 5 | **0.7903** | 0.3018 | **0.0738** |
| Epoch 6 | 0.7580 | 0.2929 | 0.0986 |
| Epoch 7 | 0.7741 | 0.2995 | 0.1488 |
| Epoch 8 | 0.7580 | 0.2785 | 0.1340 |
| Epoch 9 | 0.7580 | **0.2692** | 0.1334 |
| Epoch 10 | 0.7741 | 0.292 | 0.1497 |

| Epoch | Test Acc. | E. Risk | U. Risk |
|---|---|---|---|
| Epoch 1 | 0.7555 | 0.6206 | 0.7185 |
| Epoch 2 | 0.7555 | 0.5479 | 0.7503 |
| Epoch 3 | 0.8111 | 0.4726 | 0.5936 |
| Epoch 4 | 0.8111 | 0.3691 | 0.3939 |
| Epoch 5 | 0.7111 | 0.3252 | 0.2933 |
| Epoch 6 | 0.7 | 0.3332 | 0.2654 |
| Epoch 7 | 0.7222 | 0.286 | 0.2739 |
| Epoch 8 | **0.8111** | 0.381 | **0.2362** |
| Epoch 9 | 0.7888 | **0.2378** | 0.2617 |
| Epoch 10 | 0.7222 | 0.36 | 0.2845 |

Table 4: Evaluation of DistilRoBERTa on Persuasiveness Specificity (left) and Persuasiveness Eloquence(right)

of hidden units), that can be through topological ordering written as an upper triangular matrix $W_N^*$.

We claim that the General Insertion Algorithm can results in any upper triangular matrix of any size (depending on the number of insertion steps we do). Indeed :

- Step 4 gives the right size for the weight matrix.
- Steps 5 to 8 give us that the inserted vertical weight vector representing the links between the inserted neuron and subsequent neurons has only one constraint : the inserted neuron is not linked to himself or to other neuron on the same depth (0 connection for neurons on the same depth).
- Step 9 and 10 give us that the inserted horizontal weight vector representing the links between previous neurons and the inserted neuron has only one constraint : the inserted neuron is not linked to himself or to other neuron on the same depth (0 connection for neurons on the same depth). Those constraints prevent the network from growing cycles.
- Steps 10 to 12 give us that any previous connections can be deleted if not wanted.
- Step 13 is possible thanks to the constraints preventing the network from growing cycles and give us an upper triangular weight matrix whose only constraint is to not having a single cycle (every row is accessible with the only constraints that there are 0 for the links with neurons on previous depths).

Such a topology may not be the best with regard to recent hardware constraints. For instance, it is well-known that such sparse networks may not be easily parallelized on GPU. However, it is possible to adapt this framework if needed in order to achieve the same topology as any standard FFNN after the growing phase, such as proposed in the Constrained Insertion Algorithm detailed in the following part.

### A.3.3 Constrained Growth Algorithm

In this section, we adapt the growing general framework presented previously so that we obtain a predefined target architecture at the end of the growing process, by forcing the new neurons to be inserted directly after the input units. This constrained growing process enable us to compare strictly equivalent neural networks in terms of architectures. We add the following constraints to obtain the target architectures:

- We specify the size of the hidden layers as an input so that we can achieve a predefined architecture by growing it.
- We constrain the new neurons to be inserted directly after the input units. In this way, as soon as a layer is "completed", it is considered as an output layer and can not be modified further.
- As long as the size of the hidden layer is not attained, two new neurons cannot be inserted after the same input unit, so that only a layer is constructed at a time

We thus obtain the following algorithm, that builds a standard FFNN architecture with a chosen hidden size (which is an algorithm parameter):

**Algorithm 1** Constrained Insertion Algorithm

**Inputs**: Weights $W_t \in \mathbb{R}^{n \times n}$, $\mathcal{P}_t$ : non output neurons, $\mathcal{H}_t$ : set of already inserted neurons on this layer, $H$: Hidden Layer's desired Size ($H \leq I$), ${\mathcal{I}'}_t$: set of already used input units
**Outputs**: $W_{t+1} \in \mathbb{R}^{(n+1)\times(n+1)}$, $\mathcal{P}_{t+1}$ $\mathcal{H}_{t+1}$, ${\mathcal{I}'}_{t+1}$

Randomly sample an input node $j$ that has not already been sampled: $j \sim \mathcal{U}(\mathcal{I} \setminus {\mathcal{I}'}_t)$
Update ${\mathcal{I}'}_{t+1} = {\mathcal{I}'}_t \cup \{j\}$
Create a new node $n+1 : \mathcal{P}_{t+1} = \mathcal{P}_t \cup \{n+1\}$ and $\mathcal{H}_{t+1} = \mathcal{H}_t \cup \{n+1\}$
Initialize $W_{t+1} = \begin{bmatrix} W_t & [0]_n \\ [0]_n^T & 0 \end{bmatrix}$
Inherit children: $w_{n+1,i}^{(t+1)} = w_{j,i}^{(t)} \qquad \forall i \mid w_{j,i} \neq 0$
Link parents: $w_{i,n+1}^{(t+1)} \sim Init() \qquad \forall i \in \mathcal{I}$
Reorder $\mathcal{P}_{t+1}$ topologically so that $W_{t+1}$ is triangular
**if** $|{\mathcal{I}'}_{t+1}| = H$ **then**
  **if** $H < I$ **then**
    Remove skip-connections : $\forall (a,b) \in (\mathcal{I} \setminus {\mathcal{I}'}_{t+1}, \mathcal{P}_{t+1} \setminus \mathcal{H}_{t+1}), w_{a,b} = 0$
    ${\mathcal{I}'}_{t+1} = \emptyset$ {Start of a New Layer}
  **end if**
**end if**
**return** $W_{t+1}, \mathcal{P}_{t+1}, \mathcal{H}_{t+1}, {\mathcal{I}'}_{t+1}$

To illustrate the way this progressive insertion algorithm operates, the following Figure 1 is presented.

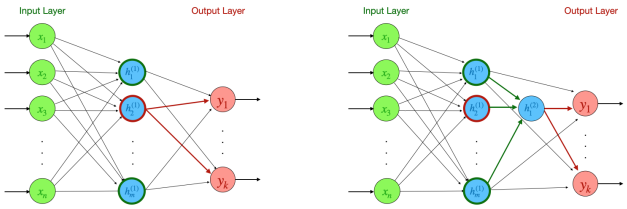


Figure 1: Example of neuron insertion: $h_2^{(1)}$ is randomly chosen and splitted to create the new neuron $h_1^{(2)}$, which inherits the output connections from $h_2^{(1)}$ (red arcs) and gets new input transitions from the previous layer (green arcs) randomly initialized.

**Algorithm 0** General Insertion Algorithm

**Inputs**: Weights $W_t \in \mathbb{R}^{\mathrm{n}\times\mathrm{n}}, \mathcal{P}_\mathrm{t}$
**Outputs**: Weights $W_{t+1} \in \mathbb{R}^{(\mathrm{n}+1)\times(\mathrm{n}+1)}$
$\mathcal{P}_{t+1}$ with $\mathcal{P}_t \subset \mathcal{P}_{t+1}$ and $|\mathcal{P}_{t+1}| = |\mathcal{P}_t| + 1$

Randomly sample a node $j$ of the network : $j \sim \mathcal{U}(\mathcal{P}_t)$
Get the sets $\mathcal{C}_j, \mathcal{X}_j, \mathcal{Q}_j$ respectively

- the sets of nodes that are children of $j$ : $\mathcal{C}_j = \{i \in \mathcal{P}_t \mid w_{j,i} \neq 0\}$.
- the sets of nodes that are topologically ordered after $j$ : $\mathcal{X}_j = \{i \in \mathcal{P}_t \mid d(i) \geq d(j) + 1\}$ (We can note that $\mathcal{C}_j \subset \mathcal{X}_j$).
- the sets of nodes that are topologically ordered before $j$: $\mathcal{Q}_j = \{i \in \mathcal{P}_t \mid d(i) \leq d(j) \wedge i \neq j\}$.

Create a new node $n+1$ and update $\mathcal{P}_{t+1} = \mathcal{P}_t \cup \{n+1\}$
Initialize $W_{t+1} = \begin{bmatrix} W_t & [0]_n \\ [0]_n^T & 0 \end{bmatrix}$ and $w_{j,n+1}^{(t+1)} = Init()$
Choose the number of children to inherit : Let Sample $\alpha = random(0, |\mathcal{C}_j|)$
Inherit some children : Sample $\alpha$ different neurons $k \sim \mathcal{U}(\mathcal{C}_j)$ and **set** $w_{n+1,k}^{(t+1)} = w_{j,k}^{(t)}$ and $w_{j,k}^{(t)} = 0$. Denote the set of these inherited nodes $\mathcal{C}_{j,\alpha}$.
Sample a random number of children on subsequent depths that are not in $\mathcal{C}_{j,\alpha}$ : Sample $\beta = random(max(0, 1 - |\mathcal{C}_{j,\alpha}|), |\mathcal{X}_j \setminus \mathcal{C}_{j,\alpha}|)$
Add new children : Sample $\beta$ different neurons $k \sim \mathcal{U}(\mathcal{X}_j \setminus \mathcal{C}_{j,\alpha})$ and set $w_{n+1,k}^{(t+1)} \sim Init()$
Sample a random number of parent on previous depths : Sample $\gamma = random(0, |\mathcal{Q}_j|)$.
Sample $\gamma$ different parent neurons $k \sim \mathcal{U}(\mathcal{Q}_j)$ and set $w_{k,n+1}^{(t+1)} \sim Init()$
Sample a random number $\delta = random(0, |\mathcal{Q}_j|)$ of $k$ neurons on previous depths that can each lose $i_k = random(0, |\mathcal{C}_k|)$ connections.
Sample $\delta$ neurons $k \sim \mathcal{U}(\mathcal{Q}_\mathrm{j})$ and $i \sim \mathcal{C}_\mathrm{k}$ connections to zero $w_{k,i} = 0$
reorder $\mathcal{P}_{t+1}$ topologically so that $W_{t+1}$ is triangular
**return** $W_{t+1}, \mathcal{P}_{t+1}$