Neural Architecture Search in complex field for audio recognition

Guibbert, Arnaud
arnaud.guibbert@epfl.ch
321085

August 18, 2024

Abstract

Neural architecture search (NAS) [1] [2] [3] find neural network architectures in an automated way. This report revisits the existing NAS problem by introducing, in addition to unary operations, pairwise operations in the search space. Those extended methods will be studied in the specific context of audio recognition. While many handcrafted efficient architectures have been already found in the context of images and NLP [4] [5] [6], NAS methods have already found different architectures that overcome them [3]. In the same way, there also exists standard architectures for audio recognition [7] that show very good performances. The objective of this project is to revisit NAS method to find efficient architectures in the audio field. More specifically, this project consists in analyzing how and where pairwise (mainly polynomial) operations could actually improve the performances. To do so, many search spaces including those operations are tackled in the NAS procedure. This report goes a step further in the NAS for audio by searching complex (as opposed to real) architectures, since the natural representation of audio is the frequency domain. The NAS is done using the differentiable architecture search (DARTS) [3] method and its variant FairDARTS [8]. In addition, a third variant, namely ForceDARTS, is presented here. Experiment results show that this new method overcomes DARTS and performs as well as FairDARTS. The introduction of pairwise operations leads to similar results, but increases the risk of generating degenerated architectures.

1 Introduction

Audio recognition is an important task in the field of audio. It goes from standard tasks such as acoustic scene classification to much harder one such as automatic speech recognition [9]. Many approaches were tackled, some works tried to extract a general embedding space [10] [11] for audio such that the recognition task could be easily performed on top of it. With the success of deep learning, deep neural networks (DNNs) [12] [7] outperformed standard machine learning algorithms such as HMM [13] or SVM [14]. While many DNN architectures have been developed in the context of image recognition (Convolutional neural networks (CNNs) [15], ResNet [4]) and NLP (Transformers [6]), these have been successfully recycled for audio recognition tasks. Nevertheless, there is less attention on NAS for audio than in the other domains, and even though audio CNN [7] and audio transformers [12] showed very good results, the architectures used may be suboptimal. Then, searching for new architectures for audio recognition is a challenging task, as it may exhibit new shapes of neural networks that could perform better than the ones recycled from image recognition and NLP tasks.

Recently, with the emergence of the neural architecture search (NAS) [16] concept, many methods have been developed to automatically search for the best architectures in neural networks. The objective of NAS is to not only optimize the weights of the network, but also optimize the architecture of this latter. Though this procedure is computationally expensive, it shows impressive results in the image field [3][17]. The NAS methods can be classified in two categories: the ones where each neural architecture is tested separately and the ones where many neural architectures are tested simultaneously, namely one-shot-NAS [18]. In the first category, algorithms such as Reinforcement learning (RL) [2] and evolutionary algorithms [19] showed good performances. They are however highly computationally demanding. The second category has the advantage to reduce the time complexity while providing competitive results. Among the most efficient techniques in one-shot NAS, the recent one DARTS [3] proposes an optimization of the architecture using gradient descent. In the wake of DARTS, many efficient variants of DARTS were released to alleviate the weaknesses of DARTS [8][20][21]. These techniques were successfully applied in the audio field and showed promising results [22][23][24]. One key parameter in DARTS (and more generally in NAS) is the search space, nevertheless most of the papers focus on the original search space proposed in [3], which is composed of unary operations (mainly convolutions). However, the expressive power of pairwise operations such as polynomials have been recently exhibited by [25], which motivated us to introduce them in the search space.

In this work, we aim at finding the best neural architectures for the supervised audio recognition task using DARTS [3] and its variant FairDARTS [8]. In addition, ForceDARTS, a variant of DARTS inspired from the approach of FairDARTS, is proposed and presented in this report. Contrary to the classical configuration of DARTS, this work explores new search spaces where polynomial and complex (as opposed to real) operations are involved. Regarding the polynomial operations, we will show to what extent such operations can be beneficial to the network and if so where should they be located. The choice of using deep complex networks [26] is motivated by the fact that the natural representation of time series is the frequency domain. While in many works, the output of the Short Time Fourrier Transform (STFT) is cast to real numbers by retaining only the magnitude, we study the effect of keeping the imaginary part and the advantage of using complex operations. In the following sections, we will refer to model using complex (respectively real) operations by **ComplexNet** (respectively **RealNet**). This paper is organized as follows: section 2 presents a short review of DARTS, FairDARTS. Weakenesses of DARTS and FairDARTS are discussed and a new method, namely ForceDARTS, that aims at alleviating them is proposed in section 3. In addition, the new diversified search spaces involving polynomial operations are presented. The descriptions of the datasets as well as the search procedure are reported in section 4. Section 5 exhibits the results obtained on the different datasets: it discusses the performances of ForceDARTS against DARTS and FairDARTS, the influence of pairwise operations and assesses the transferability of the models developed with using NAS. Our findings and conclusion are discussed in sections 6 and 7.

2 Related work

We start by reviewing some important theoretical aspects of DARTS [3] and its variant FairDARTS [8]. They are both one-shot-NAS methods, meaning that they are using a one-shot model to search the architecture. These two methods share a common structure: firstly, the architecture is searched, we will refer to it as the **search stage**. Secondly the model built with the architecture found is trained from scratch, we will refer to it as the **evaluation stage**[1]. The following sections are describing the search stage.

2.1 Differentiable architecture search (DARTS) review

This method is widely detailed in the original paper [3]. It models a neural network as a set of nodes (tensors) and edges (operations) that build a directed acyclic graph (DAG). Each edge is going from a node i to a node j. The network is constructed such that each node j is connected to all nodes i with $i < j$. Then the output of the node j reads:

$$x_j = \sum_{i<j} e_{i,j}(x_i) \tag{1}$$

Where $e_{i,j}(x_i)$ is the output of the edge going from node i to node j. For each edge $e_{i,j}$, a set of M candidate operations $O = \{o^1, o^2 ... \ o^M\}$ is defined, namely the search space. For the sake of simplicity, it assumed that all edges are sharing the same set O. The goal is to determine what is the best operation for each edge. The ideal case would be to have a one hot vector $\beta^o_{i,j} \in \{0, 1\}$ that would determine which operation is selected among the M possible ones. In DARTS, the problem is relaxed, by introducing a coefficient $\alpha^o_{i,j} \in \mathbb{R}$ that is associated with each candidate operation $o \in O$. We will denote by $\alpha_{i,j}$ the vector containing the $\alpha^o_{i,j}$ (similar definition for $\beta_{i,j}$). $\beta_{i,j}$ is then approximated by $softmax(\alpha_{i,j})$. Given an input x, the output of an edge $e_{i,j}$ reads:

$$e_{i,j}(x) = \sum_{o \in O} \frac{\exp(\alpha^o_{i,j})}{\sum_{o' \in O} \exp(\alpha^{o'}_{i,j})} o(x) \approx \sum_{o \in O} \beta^o_{i,j} o(x) \tag{2}$$

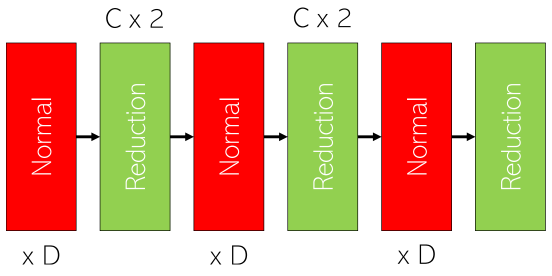
The architecture is then fully defined by the concatenation of the $\alpha_{i,j}$ $\forall i < j$. Let's denote this vector by α. Finding the best architecture boils down to finding the best α denoted α^*. To do so, we want to minimize the validation loss over α while having found the best weights $w^*(\alpha)$ for this α architecture on the training set. This can be rewritten as a bi-level optimization problem as follows:

$$\begin{aligned} &\min_{\alpha} \mathcal{L}_{val}(w^*(\alpha), \alpha) \\ &\text{s.t. } w^*(\alpha) = \arg\min\nolimits_w \mathcal{L}_{train}(w, \alpha). \end{aligned} \tag{3}$$

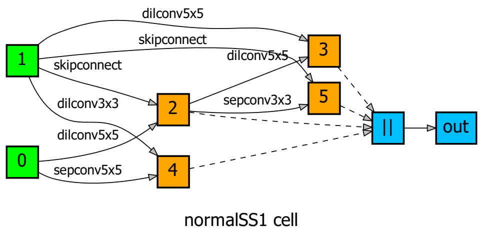
Once the best architecture, namely α^*, is found, since $\beta_{i,j}$ are approximated by the $softmax(\alpha_{i,j})$, it seems natural to select the operation with the highest coefficient. Then the selected operation for an edge $e_{i,j}$ reads:

$$o_{i,j} = \arg\max_{o \in O} \alpha^o_{i,j} \tag{4}$$

[1]It is no more, no less than the training stage of a neural network



(a)



(b)

Figure 1: The left diagram depicts the shape of the network where the architecture will be searched. It is a sequence of normal and reduction cells. The architecture is searched at the cell level, meaning that all normal cells (respectively reduction cells) are sharing the same architecture. The number of channels is increased by a factor 2 after the two first reduction cells. The figure on the right exhibits a potential architecture of a cell. This cell is parameterized by its number of intermediary nodes (orange nodes) $N = 4$, its search space and the number of input edges per nodes $k = 2$. Above each edge, the operation selected by DARTS is specified.

In practice, as done in [3], the architecture is not searched for all the edges of a network (that would result in an intractable search space). Instead, the architecture is searched at a cell level and the network is a sequence of these cells that share the same architecture, as shown on figure 1a. As done in the original paper, two types of cells are searched, namely normal and reduction cells. The first one keeps the size unchanged, while the second one divides the size by two. Each cell is characterized by its number of intermediate nodes N and each cell has two input nodes, which corresponds to the output of the two previous cells. An example of a cell with 2 input nodes and 3 intermediary nodes is shown on figure 1b.

One may notice that the node 4 on figure 1b is only connected to node 0 and 1 while definition (1) suggests that it should also be connected to node 2 and 3. Indeed, to alleviate the complexity of the cells, it is a common practice to limit the number of inputs for each node when the network is pruned. For a node j, a coefficient $\gamma_{i,j}$ is associated with each edge $e_{i,j}$. The output of a node j is redefined as follows:

$$x_j = \sum_{i<j} \frac{\exp(\gamma_{i,j})}{\sum_{i<j} \exp(\gamma_{i,j})} e_{i,j}(x_i) \tag{5}$$

Again, we will denote by γ_j the vector containing the $\gamma_{i,j}$ $\forall i < j$ and by γ the concatenation of the γ_j. We will refer to γ as the edge weights. γ is now part of the architecture search and optimized in the same way it is done with α. Once γ is optimized, given a node j, we only retain the k edges with the highest $\gamma_{i,j}$. This k is usually set to 1 or 2, in this work it is set to 2. The configuration used for DARTS (search space, number of nodes ...) is detailed in the experimental set-up section.

2.2 FairDARTS review

Many variants of DARTS were proposed: among them, FairDARTS [8] gave very good results. In the paper, they highlight two reasons why the performances of DARTS may collapse and proposed two modifications to avoid such a drop.

In their work, they first notice that the "skip connection" candidate operation was benefiting from an unfair advantage in the original configuration of DARTS. Indeed, the way the candidate operations are aggregated (equation 2) is close to a ResNet structure [4]. Then the "skip connection" takes advantage of parallel connections when the architecture is searched, which increases its architectural weight. Since the *softmax* function is used, if the architectural weight of the skip connection increases, this automatically causes the other architectural weights to decrease. Since DARTS only retains the best operation, this finally leads to a poor architecture where "skip connections" are dominant. In [8], this phenomenon is called the **unfair advantage** of skip connections.

The second issue noticed in [8] is the uncertainty regarding the choice of the retained operation.

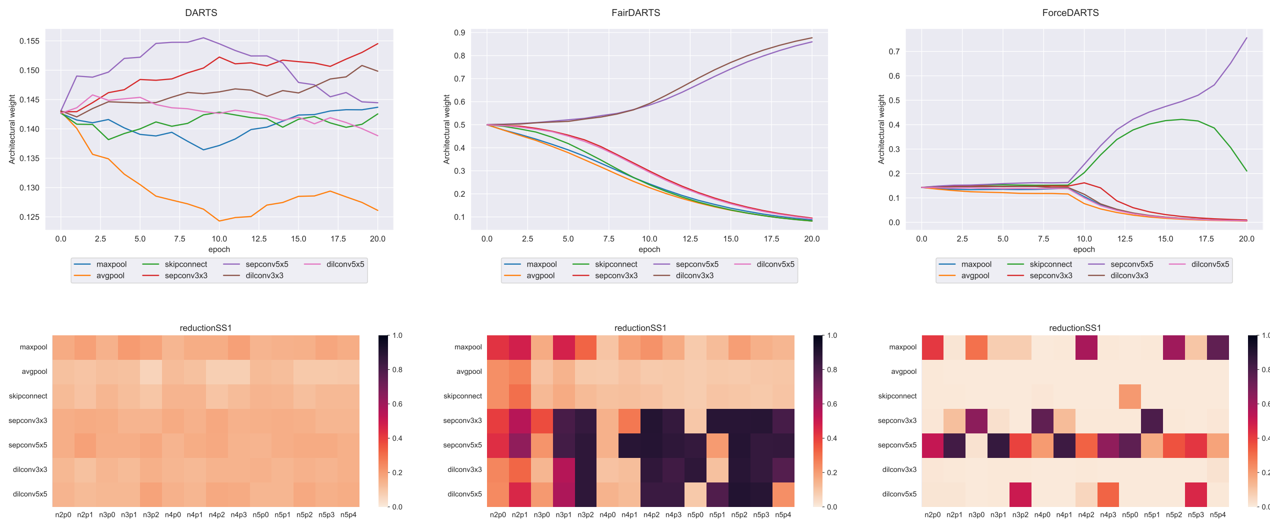


Figure 2: These figures are depicting the behavior of the architectural weights for DARTS (left), FairDARTS (middle) and ForceDARTS (right). These plots are coming from experiments on the SC09 dataset. The line plots are depicting the evolution of the architectural weights for the edge 0-5 of the reduction cell, while the heat maps show the final values of the architectural weights for each operation (y-axis) and for each edge (x-axis). The final architectural weights learned by DARTS are very close to each other, such that the final choice is uncertain. While the final architectural weights are not a close to a one-hot structure, FairDARTS is exhibiting a subset of "good" operations and reduces the uncertainty. Finally, ForceDARTS exhibits at most two "good" operations and provides architectural weights that are very close to a one-hot structure.

Indeed, it can be noticed that the final α found by DARTS are such that we are far from the approximation stated in equation (2), namely that $softmax(\alpha_{i,j})$ is close to a one hot vector. Such a result is shown on figure 2. It results that the final choice is somehow uncertain. Moreover, the final discretization step (pruning) will then lead to a very different network and the performances would then potentially drop as suggested in [20].

FairDARTS first alleviates the **unfair advantage** of skip connections by replacing the $softmax$ function by the $sigmoid$ such that equation (2) becomes:

$$e_{i,j}(x) = \sum_{o \in O} \sigma(\alpha^{o}_{i,j}) o(x)$$

Such a transformation will not prevent skip connections from having high architectural weights, but the weights of the other candidate operations will no longer be directly affected by the increase of skip connections architectural weights. At the end, all the "good" operations should have a high architectural weight (i.e. high $\sigma(\alpha^{o}_{i,j})$). Contrary to the initial **one-hot-structure** suggested in DARTS, FairDARTS propose a **multi-hot-structure** where many operations can be selected.

The second issue related to the final uncertainty is actually solved by introducing a regularization loss in the validation loss function from equation (3). Such a loss aims at pushing the architectural weights either towards 0 or 1 (maximizing the certainty). The new regularized validation loss $\hat{\mathcal{L}}_{val}$ is then equal to:

$$\hat{\mathcal{L}}_{val} = \mathcal{L}_{val} + w_{0-1} L_{0-1}$$
$$\text{with} \quad L_{0-1} = -\frac{1}{N} \sum_{i}^{N} (\sigma(\alpha_i) - 0.5)^2$$

Where w_{0-1} is the weight associated with the loss L_{0-1}. Since FairDARTS provides a **multi-hot-encoding**, multiple operations can be retained as far as they are beyond a given threshold, i.e. $\sigma(\alpha_i) > \sigma_{threshold}$. In addition to $\sigma_{threshold}$, a hard limit is introduced such that, only p operations are at most retained. In this work we will set $p = 1$. All the modifications discussed previously are also applied in the same way to the edge weights γ.

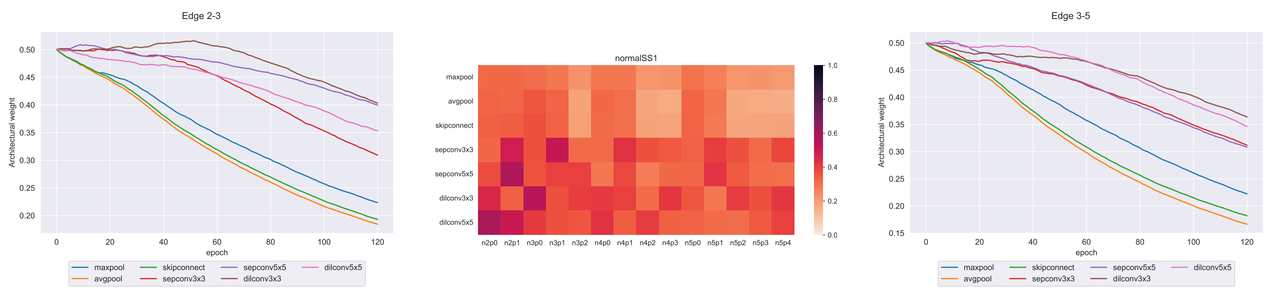


Figure 3: These plots are coming from FairDARTS experiments on the ESC50 dataset. The line plots are depicting the evolution of the architectural weights for the edge 2-3 (left) and 3-5 (right) of the normal cell, while the heat map shows the final values of the architectural weights for each operation (y-axis) and for each edge (x-axis). One can notice that the final architectural weights for a given edge are close to 0 while one would expect that at least one operation will have a weight close to 1 (selected operation). The final choice is then made by selecting the "least ill" operation.

3 Methods

3.1 ForceDARTS

Even though FairDARTS alleviates some weaknesses of DARTS, it introduces some undesirable artifacts. The replacement of the *softmax* function by the *sigmoid* one allows each operation to be independent of the others, and because of this independence, it does not prevent them to all go to 0 or 1. Such a behavior is depicted in figure 3 where all architectural weights are decreasing towards 0. Then does it make sense to retain the operation with the highest architectural weight when they are all close to 0 ? We are back to the uncertainty problem. Besides, figure 3 exhibits the fact that using *sigmoid* function instead of *softmax* may increase the gap between the one-shot-model and the pruned model.

The second weakness is related to the selection of the input edges for each node in the cell (1b). This point is not widely discussed in DARTS [3] as well as in [8], although it raises some inconsistencies. Each node has a limited number of input edges, namely k, thus one would expect to have a multi-hot structure for γ_j, where k weights would be equal to 1 and the others equal to 0. However, DARTS cannot provide such a structure since it uses the *softmax* function and FairDARTS can potentially reach such a structure but does not impose it. One may end up with edge weights such as those described in figure 4. Having such edge weights increases the gap between the one-shot-model and the pruned model.

So far, we highlighted the fact that in both DARTS and FairDARTS the gap between the one-shot model and the pruned model may be high, but what is the underlying problem with such a gap ? Actually following the intuition introduced in [20], the higher the gap between the optimal one-shot model and pruned one, the higher the risk to get worse performances with the pruned model. In this work, The gap between the one shot model M_{os}, and the pruned model M_p, will be defined as follows:

$$\| M_{os} - M_p \| = \frac{1}{|e_r| + |n|}\Big(\sum_{e_{i,j} \in e_r} d(\alpha^{os}_{i,j}, \alpha^{p}_{i,j}) + \sum_{n_j \in n} d(\gamma^{os}_j, \gamma^{p}_j) \Big)$$
$$\text{with } d \text{ defined as} \quad \forall u, v \in \mathbb{R}^K \quad d(u,v) = \frac{1}{K} \| u - v \|_1 \tag{6}$$

Where, e_r is the set of retained edges and n the set of intermediate nodes. This gap distance is by definition included in $[0, 1]$. ForceDARTS aims at reducing the gap between M_{os} and M_p by softly forcing the algorithm to choose the architecture among the ones that are closed to a one-hot structure for the architectural weights and multi-hot structure for the edge weights.

Pushing the architectural weights towards a pure distribution

The discrepancy of discretization is solved in FairDARTS by pushing the architectural weights either towards 0 or 1, but it does not explicitly impose them to reach a one-hot structure. To impose this structure, we will first go back to the original $softmax$ function (instead of $sigmoid$ one) for the architectural weights. The intuition behind ForceDARTS is also to introduce a loss that will force the architectural distribution of an edge ($softmax(\alpha_{i,j})$) to become a **pure** distribution (i.e. one hot vector as expected). Given a distribution $p = \{p_1, p_2 ... \, p_C\}$, one would like to find a loss that is maximal when the distribution is uncertain (impure, i.e $p_i = p_j \; \forall i, j$) and minimal when the distribution is a one hot vector. Such losses have already been defined in the context of decision trees where two functions are mainly used to measure the purity of a distribution, namely the **Gini impurity loss** $Gini$ and the **Entropy** E. Given a discrete distribution p, those two losses are defined as follows:

$$Gini(p) = \sum_{i=1}^{C} p_i(1-p_i) \quad E(p) = -\sum_{i=1}^{C} p_i \log(p_i) \quad \text{s.t.} \quad \sum_{i=1}^{C} p_i = 1 \tag{7}$$

With the convention that $0\log(0) = 0$. They both reach their minimum value (namely 0) for a pure distribution. They reach their maximum value ($(C-1)/C$ for the Gini impurity and $\log(C)$ for the Entropy) for an impure distribution. Unfortunately, those two functions are concave and have as many global minimums as the length of the distribution, namely C. Then adding such a loss may cause the following artifact: the operations would be rather selected such that one of the global minimum of the regularization loss is reached than for their contribution to the architecture. Hence, if this loss is too large against the original loss, the evolution of the architectural weights would be strongly initialization-dependent.

To solve this issue, two options are considered. The first one consists in carefully setting the multiplicative weight w_{purity} in front of the regularization loss. One can imagine a weight that would increase with the number of epochs. A second approach, that will be used in this paper, consists in disabling this loss until a certain number of epochs e_c and then enable it with a high weight w_{purity}. The intuition behind this approach is to first let the original loss guide the network towards a potential architecture (without forcing it such that it is not initialization-dependent) and once it is sufficiently closer to a global minimum, force the architectural weights to go inside it (become pure). In this work, the loss is disabled until the half of the training stage and then enabled with $w_{purity} = 20$. To make the functions defined in 7 distribution-agnostic, they are normalized between $[0, 1]$. The new regularized validation loss $\hat{\mathcal{L}}_{val}$ is now equal to:

$$\hat{\mathcal{L}}_{val} = \mathcal{L}_{val} + w_{purity} f_{purity}(\alpha) \times \mathbb{1}_{(epochs > e_c)}$$

$$\text{with} \quad f_{purity}(\alpha) = \frac{1}{|e|} \sum_{e_{i,j} \in e} \mathcal{L}_{purity}(softmax(\alpha_{i,j}))$$

Where e is the set of edges and $\mathcal{L}_{purity}$ is either $Gini$ or E. In this work, the Shannon entropy loss will be used. As shown on figure 2, the distributions obtained using ForceDARTS are close to pure distributions.

Pushing edge weights towards a multi-hot structure

As recalled before, k input edges are selected for each edge. Then one is actually looking for a multi-hot structure where k elements are equal to 1 and the rest equal to 0. Hence, using the $softmax$ function for the edge selection, as defined in equation (5), will not lead to the expected structure. As done in FairDARTS the $softmax$ function will be replaced by the $sigmoid$ function in the edge selection such that equation (5) becomes:

$$x_j = \sum_{i<j} \sigma(\gamma_{i,j}) e_{i,j}(x_i)$$

The second step consists in pushing the edge weights towards a multi-hot structure where k elements are equal to 1 and the rest equal to 0. Given a sequence $\{p_1, p_2 \; ... \; p_L\}$ to be pushed towards a multi-hot structure, a transformation will first be applied such that we generate a probability distribution q from $\{p_1, p_2 \; ... \; p_L\}$. Without loss of generality, we assume $k = 2$ for the next steps, then q is defined $\forall i \neq j$ as follows:

$$\forall i \neq j \quad q_{i,j} = \frac{p_i p_j}{\sum_{k \neq l} p_k p_l}$$

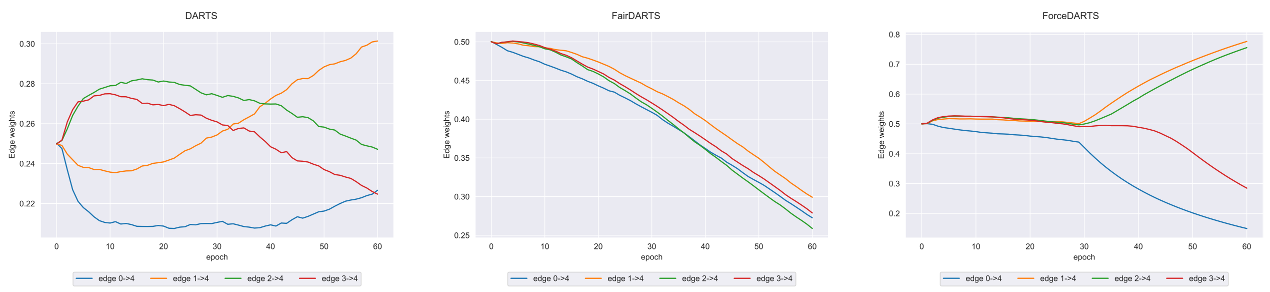


Figure 4: These plots are coming from experiments on SC09. The line plots are depicting the evolution of the edge weights for the node 4 of the reduction cell for DARTS (left), FairDARTS (middle) and ForceDARTS (right). At the end, the algorithm retains two edges. As explained before, DARTS as well as FairDARTS do not reach the expected multi-hot structure. The result of FairDARTS is very poor, as all the edge weights are converging towards 0. The introduction of the additional loss in ForceDARTS allows reaching the expected structure and then reduces the gap between the one-shot-model and the pruned model.

Assuming $p_i \in [0,1]$, there is an equivalence between the multi-hot structure of $\{p_1, p_2 \ldots p_L\}$ and the purity of the distribution q. In mathematical terms:

$$q \text{ is pure} \quad \Leftrightarrow \quad \exists\, k, l \quad \text{with} \quad k \neq l \quad \text{s.t.} \quad p_k = p_l = 1 \quad \text{and} \quad \forall i \neq k, l \quad p_i = 0$$

The problem then boils down to push q towards a pure distribution. Given a node j, the associated loss $\mathcal{L}_{multihot}$ is then equal to:

$$\mathcal{L}_{multihot}(\gamma_j) = \mathcal{L}_{purity}(q) \quad \text{with} \quad q_{k,l} = \frac{\gamma_{k,j}\gamma_{l,j}}{\sum_{u \neq v} \gamma_{u,j}\gamma_{v,j}}$$

Where $\mathcal{L}_{purity}$ is either $Gini$ or E. The new final regularized loss $\hat{\mathcal{L}}_{val}$ of the ForceDARTS method is then equal to:

$$\hat{\mathcal{L}}_{val} = \mathcal{L}_{val} + (w_{purity} f_{purity}(\alpha) + w_{multihot} f_{multihot}(\gamma)) \times \mathbb{1}_{(epochs > e_c)}$$

$$\text{with} \quad f_{purity}(\alpha) = \frac{1}{|e|} \sum_{e_{i,j} \in e} \mathcal{L}_{purity}(softmax(\alpha_{i,j})) \quad \text{and} \quad f_{multihot}(\gamma) = \frac{1}{|n|} \sum_{n_j \in n} \mathcal{L}_{multihot}(\gamma_j)$$

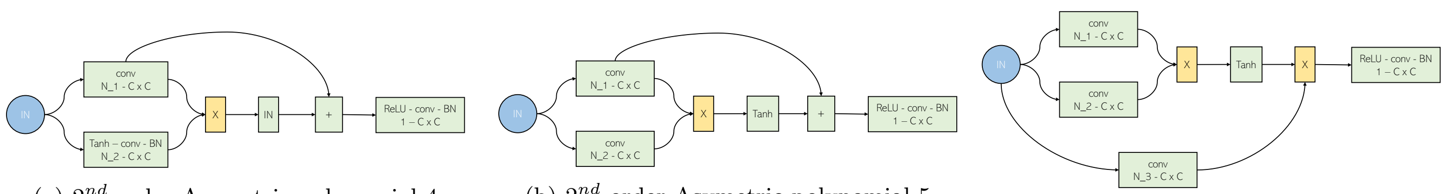
Where n is the set of intermediary nodes in a cell. We set $w_{multihot} = w_{purity} \times |n|/|e|$ such that nodes and edges are weighted equally.

3.2 Diversification of the search space

Beyond the method itself, the existing NAS methods can be improved by optimizing existing parameters of the method. As recalled before, most of the existing works involving DARTS do not focus on the search space and recycle the original one presented in [3]. In this work, the search space is diversified by introducing polynomial operations. The final objective is to determine how such operations could improve the final results. Let's first redefine the search space proposed in the original paper [3]. The search space consists of the 7 following candidate operations:

- Separable convolutions 3x3 and 5x5
- Dilated convolutions 3x3 and 5x5
- Max pooling 3x3, Average pooling 3x3 and skip connection

In the rest of the report, this search space will be denoted **SS1**. This latter will be used as a baseline for the other search spaces that will be explored. Among many search spaces that have been tested, three different other search spaces will be presented here. To assess the competitiveness of polynomial operations w.r.t. convolutions, the new search spaces will be generated such that polynomial operations will only replace separable convolution 5x5 and dilated convolution 5x5. Hence, they will be in competition with separable convolutions 3x3 and dilated convolutions 3x3. The search spaces are defined as follows:



(a) 2^{nd} order Asymetric polynomial 4

(b) 2^{nd} order Asymetric polynomial 5

(c) 3^{rd} order Attention polynomials

Figure 5: Structure of different polynomial operations, C is the number of channels. On the left and middle, the Asymetric polynomials 4 and 5 are both parameterized by the kernel size of the first convolution N_1 and the size of the kernel of the second convolution N_2. On the right, the attention polynomials are parameterized by N_1, N_2 and N_3 respectively the kernel sizes of the first, second and third convolution operations. To limit the number of parameters of the operations defined above, the convolutions involved in a product are using as many groups as channels C.

- **SSPolAsym4** : the two convolutions are replaced by Asymetric polynomials 4 (structure depicted on figure 5a). The first new operation is an Asymetric polynomial 4 with $N_1 = 7$ and $N_2 = 1$ and the second one is an Asymetric polynomial 4 with $N_1 = 5$ and $N_2 = 3$.

- **SSPolAsym5** : the two convolutions are replaced by Asymetric polynomials 5 (structure depicted on figure 5b). The first new operation is an Asymetric polynomial 5 with $N_1 = 7$ and $N_2 = 1$ and the second one is an Asymetric polynomial 5 with $N_1 = 5$ and $N_2 = 3$.

- **SSPolAttention** : the two convolutions are replaced by Attention polynomials (structure depicted on figure 5c). The first operation is an Attention polynomial with $N_1 = 7$, $N_2 = 1$ and $N_3 = 1$, the second one is an Attention polynomial with $N_1 = 5$, $N_2 = 3$ and $N_3 = 3$.

Many other polynomials operations were considered, however their performances are not presented here since they were mainly not selected in the final architecture. One may wonder why convolution kernels are relatively high in the new operations added. Those high kernels are introduced in order to break the symmetry between the two tensors involved in the final product. We empirically observed that the higher the asymmetry, the more likely the polynomial operation will be chosen. Besides, the Asymetric polynomial 4 structure, depicted on figure 5a, is designed such that the asymmetry is strengthened by the non-linearity introduced in the second branch. Following the intuition from [25], a skip connection branch has been added in both Asymetric polynomials 4 & 5 configurations to combine the first and second order. Finally, the third architecture, namely Attention polynomials, outputs a third order tensor and was mainly inspired from the transformers [6] architecture.

4 Experimental set-up

4.1 Datasets and pre-processing

To assess the performances of the methods described below, four audio datasets were chosen:

- Two "easy" and similar datasets called **AudioMNIST** [27] and **SC09** [28]. As its name suggests, the first one is an equivalent of **MNIST** for the audio domain. The second one is also a dataset of spoken digits, but it is actually a subset of the **SC35** [28] dataset that contains other spoken words.

- An intermediary dataset called **SC35** [28]. This dataset consists of 35 classes of spoken words.

- A much harder dataset called **ESC50** [29]. This dataset consists in 50 classes of environmental audio recordings. This dataset is much harder since it has only few samples.

While our methods will be assessed on SC09, SC35 and ESC50, AudioMNIST will only be used to evaluate the transferability of the models learned. The pre-processing pipeline is mainly depicted on figure 6. The first data augmentation performed on the raw audio consists in randomly modifying the amplitude of the signal and randomly fading it. Then the signal is encoded into the frequency domain by sequentially applying the short time Fourier transform (STFT), Mel filters and a logarithmic transformation. In parallel, the same transformations are applied to a noise audio data and this latter is added to the signal after the logarithmic transformation.

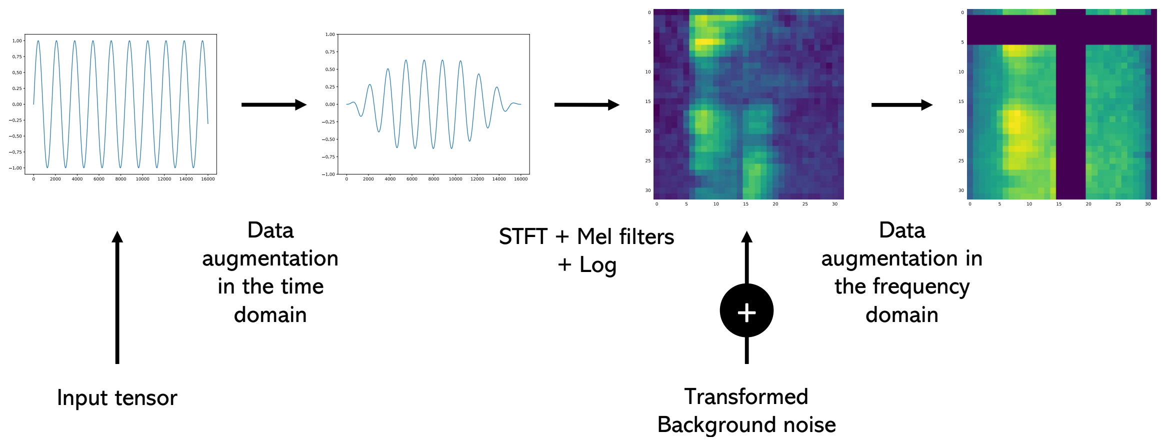

Data augmentation in the time domain
STFT + Mel filters + Log
Data augmentation in the frequency domain
Input tensor
Transformed Background noise

Figure 6: The figure exhibits the pre-processing pipeline for an audio input. A first data augmentation is performed in the time domain. Then the short time Fourier transform as well as Mel filters are applied. Finally, a background noise on which the same transformation is applied is added to the spectrogram and a last data augmentation is performed on the resulting image.

Finally, a last data augmentation is performed in the frequency domain: the time is randomly stretched, columns and rows are randomly masked. On the top of that, Mixup [30] transformation is performed only in the evaluation stage. The other characteristics are reported in table 1.

Dataset	Size	train set (K)	test set (K)	classes
AudioMNIST	1x32x32	21	4.5	10
SC09	1x32x32	18.62	2.55	10
SC35	1x32x32	84.84	11	35
ESC50	1x56x146	1.4	0.3	50

Table 1: Characteristics of each dataset. The **Size** column corresponds to the size of the audio spectrogram after the pre-prossessing step.

4.2 Search stage

As recalled before, this is the stage where the optimal α^* and γ^* are searched, that is to say, the bi-level optimization defined in equation (3) is solved. Due to the GPU memory limit and in relation with the dataset difficulty, the number of channels, the number of nodes per cells as well as the number of cells must be chosen carefully for each dataset. For each one, the experimental set-up of the search stage is defined in table 2.

Regarding, the train/validation split for the search stage, the initial training set is halved such that the first part is used to train the weights and the second part to train the architecture. Regarding the parameters related to the optimization procedure, the first order method described in [3] (i.e. second order disabled) is used. As in [3], the architectural weights α and γ are trained with Adam using an initial learning rate $\eta = 3 \times 10^{-4}$, a momentum $\beta = (0.5, 0.999)$ and a weight decay of 10^{-3}. The weights are trained using momentum SGD with initial learning rate $\eta_w = 0.025$, momentum of 0.9, and weight decay of 3×10^{-4}.

4.3 Evaluation stage

The evaluation stage consists in training from scratch the model built using the architecture found during the search stage. The number of normal cells as well as the number of channels are increased such that the model is sufficiently big to handle the task. These values are reported in table 2.

Regarding the training procedure, The weights are trained in the same way it is done in the search stage. In addition, a cosine annealing learning rate scheduler without restart is used. The lower bound of this scheduler has been set to 10^{-5}.

Dataset	Type	Phase	Nodes	Channels	Normal cells	Parameters(M)	batch size	epochs
SC09	Complex	search	3	14	1	1.67	96	60
		retrain	3	22	4	_	128	200
SC09	Real	search	3	16	2	1.61	96	60
		retrain	3	26	6	_	128	200
SC35	Complex	search	4	14	1	2.87	72	20
		retrain	4	22	4	_	128/96*	150/120*
SC35	Real	search	4	18	2	3.04	72	20
		retrain	4	26	6	_	128	150
ESC50	Complex	search	4	14	1	2.87	12/8*	120
		retrain	4	22	5	_	16/10*	450
ESC50	Real	search	4	18	2	3.04	12	120
		retrain	4	26	8	_	16	450

Table 2: The number of parameters is computed using the **SS1** search space, nevertheless other search spaces are designed such that they have approximately the same number of parameters as **SS1**. Elements marked up with a star are the parameters used for the polynomials search spaces. Indeed, due to the higher complexity of polynomials architectures and to the re-implementation of complex operations, the batch size has to be re-adjusted for those search spaces.

5 Results

The following results were obtained using a Tesla V100. The methods described in section 3 were implemented using the NNI [31] library. The code is made available at this github.

5.1 Searching with ForceDARTS

First, the relevance of the new method presented in 3, namely ForceDARTS, is assessed. To do so, we compare it with the two existing methods DARTS and FairDARTS on SC09, SC35 and ESC50 using the original search space, namely **SS1**. The results are both reported in table 3 and in the bar plots from figure 8. The best architectures found are depicted on figure 7.

Overall, the performances provided by FairDARTS and ForceDARTS methods are better than the ResNet18 baseline on SC35 and SC09, and they are a little bit lower on ESC50. This is mainly due to the fact that the number of parameters[2] is too small to be able to reach similar performances.

[2]due to the complex architectures and NNI module, it was not possible to go beyond 5M parameters on our GPU.

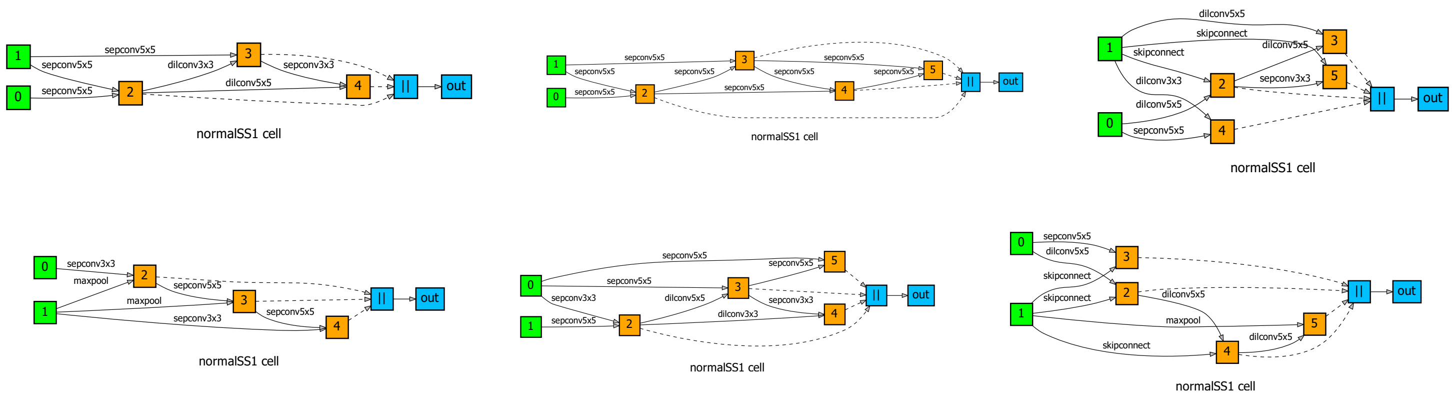


Figure 7: The diagrams are showing the two best architectures found on each dataset using the **SS1** search space. The best ones found on SC09 are depicted in the left column, the middle one corresponds to SC35 and the right one exhibits the best architectures learned on ESC50. The top row corresponds to the architectures giving the best performance, and the bottom one corresponds to the architectures giving the second best results.

Dataset	Model	Method	Accuracy(%)		Params(M)		Gap		time(h)	
			A	B	A	B	A	B	A	B
SC09	ComplexNet	DARTS	97.43 ± 0.05	96.85 ± 0.12	1.45	2.05	0.29	0.28	5	6
		FairDARTS	**97.86 ± 0.03**	97.77 ± 0.02	2.97	2.88	0.44	0.43	10	10
		ForceDARTS	97.81 ± 0.06	97.68 ± 0.05	2.85	2.32	0.18	0.18	10	8
	RealNet	DARTS	97.24 ± 0.11	97.31 ± 0.07	1.67	1.49	0.29	0.29	2	2
		FairDARTS	97.77 ± 0.1	97.71 ± 0.06	2.61	2.52	0.44	0.41	2	2
		ForceDARTS	97.53 ± 0.06	**97.86 ± 0.04**	2.01	2.25	0.18	0.18	2	2
	ResNet18	_	97.77 ± 0.04	_	11.12	_	_	_	1.2	_
SC35	ComplexNet	DARTS	96.53 ± 0.05	97.06 ± 0.04	3.42	3.76	0.28	0.27	37	44
		FairDARTS	97.33 ± 0.06	97.1 ± 0.03	4.38	4.58	0.38	0.39	46	49
		ForceDARTS	97.17 ± 0.05	**97.36 ± 0.02**	4.77	4.75	0.1	0.1	52	52
	RealNet	DARTS	96.76 ± 0.02	97.05 ± 0.06	3.19	3.25	0.28	0.28	11	14
		FairDARTS	**97.34 ± 0.05**	97.28 ± 0.04	4.29	4.61	0.36	0.38	15	17
		ForceDARTS	97.02 ± 0.11	97.15 ± 0.03	3.73	3.8	0.09	0.09	12	12
	ResNet18	_	95.91 ± 0.06	_	11.12	_	_	_	4.6	_
ESC50	ComplexNet	DARTS	45.22 ± 4.49	79.78 ± 1.4	2.67	2.91	0.26	0.26	11	11
		FairDARTS	**84.44 ± 0.55**	83.44 ± 0.64	3.85	3.63	0.41	0.41	14	12
		ForceDARTS	80.0 ± 0.41	83.0 ± 0.36	3.85	3.09	0.2	0.2	16	11
	RealNet	DARTS	80.22 ± 1.14	46.22 ± 9.06	3.55	1.63	0.27	0.26	3	2
		FairDARTS	77.33 ± 0.72	80.67 ± 0.49	3.07	2.14	0.4	0.4	3	2
		ForceDARTS	79.78 ± 0.89	81.89 ± 0.28	2.84	2.23	0.21	0.19	3	2
	ResNet18	_	**86.00 ± 0.41**	_	11.12	_	_	_	0.7	_

Table 3: The architecture is searched two times. The architectures found are respectively called A and B. For each architecture found, the evaluation stage is repeated three times with different seeds. The accuracy reported is the mean test accuracy at the evaluation stage, and for each dataset the two best performances are highlighted. The time reported in the table corresponds to the time it takes to train the pruned model in the evaluation stage. The gap entry corresponds to the gap defined in (6) between the one-shot-model and the pruned model. All the experiments reported in this table were obtained using the **SS1** search space.

Our new approach ForceDARTS overcomes DARTS and provides performances comparable to FairDARTS ones. Indeed, for the 6 proposed configurations:

- the best (respectively worst) performance[3] of ForceDARTS is always better than the best performance (respectively worst) provided by DARTS.
- the worst performance of ForceDARTS is better than the best performance of DARTS 4 times out of 6.
- the best performance of ForceDARTS is better than the best one of FairDARTS 3 times out of 6.
- the worst performance of ForceDARTS is better than the worst one of FairDARTS 2 times out of 6.

Besides, ForceDARTS reaches **97.36%** on SC35 and **97.86%** on SC09, that are the best performances on those datasets. It provides lower performances on ESC50 reaching **83.0%** that is **1.5%** less than the best performance provided by FairDARTS. One can also notice that DARTS provides degenerated architectures on ESC50 2 times out of 4, while such a behavior is not observed for ForceDARTS and FairDARTS, showing a higher robustness for those latter.

On the top of this, one can also remark that FairDARTS as well as ForceDARTS tends to exclude parameter-less operations. For instance, on SC35, FairDARTS (resp. ForceDARTS) provides complex architectures that have on average **4.48 M** (resp. **4.76 M**) against **3.59 M** for DARTS. Overall, architectures found by ForceDARTS have 1.3 times more parameters than the ones found by DARTS. While such a behavior was explicitly expected for FairDARTS, it is quite surprising to figure out that ForceDARTS does not suffer from the unfair advantage phenomenon highlighted in

[3]the best accuracy between architecture A and B.

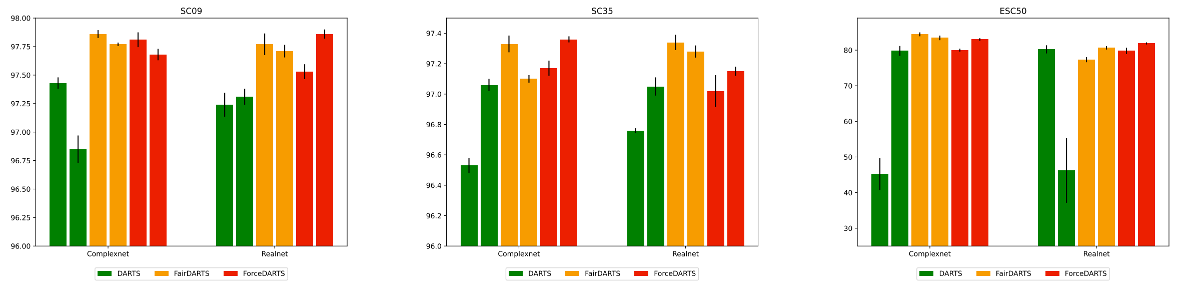


Figure 8: The mean accuracy at the evaluation stage of the three methods are reported in the above bar plots (left SC09, middle SC35 right ESC50). The error bars are exhibiting the standard deviations. For each method, two bars are plotted, representing the performances of the architectures A and B. These results are obtained using the **SS1** search space.

[8] while using the $softmax$ function. This could be explained by the fact that ForceDARTS auto-regularizes itself by getting closer to one-hot structures. Let's consider a structure where the skip connection is in competition with another operation: at the beginning, the skip connection benefits from the unfair advantage coming from the other operations. However, as soon as the structure tends to a one-hot-vector, the architectural weights of the other operations drop to 0 and the skip connection loses its unfair advantage. The operation in competition with the skip connection is then selected. Such a behavior is depicted on figure 2. Finally, regarding the gap between the one-shot-model and the pruned model, as expected, ForceDARTS provides the lowest gap while FairDARTS provides higher gap than DARTS.

5.2 Benefits of polynomials

The influence of the introduction of polynomial operations in the search space is presented here. Due to computational reasons, the three search spaces presented in 3 cannot be evaluated using the three NAS different methods. They will rather be compared using FairDARTS as this method has already proved its worth. The results are both reported in table 4 and bar plots of figure 9. The best architectures involving polynomial operations are depicted on figure 10.

Even though polynomial operations are chosen against unary operations, overall they do not highly improve the results found with **SS1** search space. Furthermore, the architectures found with polynomial candidate operations are less robust than the ones found by **SS1**. This statement particularly holds for complex architectures[4]: using **SSPolAsmy4** 5 complex architectures out of 6 are degenerated, and using **SSPolAsmy5** 3 complex architectures out of 6 are degenerated. Regarding real architectures, only one is degenerated.

[4]architectures involving complex operations.

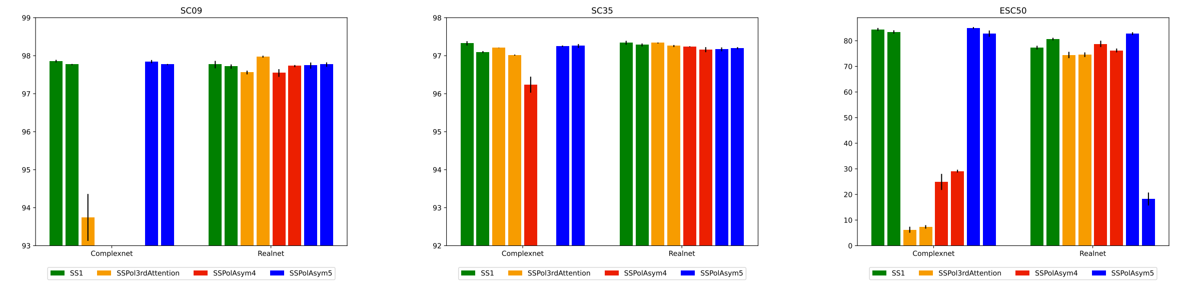


Figure 9: The mean accuracy at the evaluation stage of the four different search spaces are reported in the above bar plot (left SC09, middle SC35, right ESC50). The error bars are exhibiting the standard deviation. For each search space two bars are plotted, representing the performances of architecture A and B. To be able to distinguish the difference between the top performances, the y-axis scale has been adapted. A blank space means that the accuracy lies below the lower bound of the y-axis. These results are obtained using FairDARTS in the search phase.

Dataset	Model	Search Space	Accuracy(%)		Params(M)		Polynomials(%)		time(h)	
			A	B	A	B	A	B		
SC09	ComplexNet	SS1	**97.86 ± 0.03**	97.77 ± 0.02	2.97	2.88	0	0	10	10
		SSPol3rdAttention	93.74 ± 0.62	40.82 ± 3.58	2.24	2.66	25	41	9	12
		SSPolAsym4	42.71 ± 2.93	22.49 ± 1.91	2.51	2.43	66	66	13	14
		SSPolAsym5	97.84 ± 0.04	97.77 ± 0.02	2.43	2.45	33	33	10	11
	RealNet	SS1	97.77 ± 0.1	97.71 ± 0.06	2.61	2.52	0	0	2	2
		SSPol3rdAttention	97.56 ± 0.05	**97.98 ± 0.02**	2.1	2.36	41	50	3	3
		SSPolAsym4	97.54 ± 0.1	97.73 ± 0.03	2.48	2.39	75	75	4	4
		SSPolAsym5	97.74 ± 0.08	97.77 ± 0.06	2.63	2.38	41	33	3	3
SC35	ComplexNet	SS1	97.33 ± 0.06	97.1 ± 0.03	4.38	4.58	0	0	46	49
		SSPol3rdAttention	97.21 ± 0.0	97.02 ± 0.01	4.15	4.11	6	6	41	42
		SSPolAsym4	96.24 ± 0.21	61.8 ± 11.7	4.09	3.62	50	62	51	52
		SSPolAsym5	97.25 ± 0.02	97.26 ± 0.05	3.94	4.11	18	37	44	48
	RealNet	SS1	**97.34 ± 0.05**	97.28 ± 0.04	4.29	4.61	0	0	15	17
		SSPol3rdAttention	**97.33 ± 0.01**	97.26 ± 0.03	3.72	3.68	25	18	15	13
		SSPolAsym4	97.24 ± 0.01	97.16 ± 0.07	3.45	3.53	87	68	22	20
		SSPolAsym5	97.17 ± 0.05	97.2 ± 0.03	3.64	3.56	43	81	16	18
ESC50	ComplexNet	SS1	**84.44 ± 0.55**	83.44 ± 0.64	3.85	3.63	0	0	14	12
		SSPol3rdAttention	6.22 ± 1.17	7.33 ± 0.68	4.27	4.43	81	93	35	39
		SSPolAsym4	24.89 ± 3.14	29.11 ± 0.55	4.2	3.93	100	87	40	35
		SSPolAsym5	**85.0 ± 0.36**	82.78 ± 1.23	4.18	3.07	12	12	23	18
	RealNet	SS1	77.33 ± 0.72	80.67 ± 0.49	3.07	2.14	0	0	3	2
		SSPol3rdAttention	74.44 ± 1.23	74.56 ± 0.91	3.55	4.13	81	100	8	9
		SSPolAsym4	78.78 ± 1.23	76.22 ± 0.75	3.32	3.55	68	81	8	8
		SSPolAsym5	82.78 ± 0.55	18.22 ± 2.53	3.16	1.73	31	6	3	2

Table 4: The architecture is searched two times. The architectures found are respectively called A and B. For each architecture found, the evaluation stage is repeated three times with different seeds. The accuracy reported is the mean test accuracy at the evaluation stage, and for each dataset the two best performances are highlighted. The time reported in the table corresponds to the time it takes to train the pruned model in the evaluation stage. The **Polynomials** column corresponds to the percentage of polynomial operations in the architecture found. All the experiments reported in this table were obtained using FairDARTS.

It then suggests that the product operation in the complex field increases the risk of overfitting and the instability of the network. Moreover, except on ESC50, the best performances with polynomials are reached using real neural network: **97.33%** against **97.26%** on SC35 and **97.98%** against **97.84%** on SC09. On ESC50, the best performance is obtained using **SSPolAsym5** search space and it outperforms the architecture found using **SS1**.

It seems that polynomial operations behave differently in the complex field than in the real field. We took a deeper look on the conditions of appearance of degenerated architectures. The relation between the normalized accuracy and the percentage of polynomial operations involved in the architecture found is depicted on figure 11. One can first notice that the two quantities are, a priori, not correlated for the real case. Regarding the complex case, negative slopes are observed, suggesting that the higher the density of polynomial operations, the lower the final performances. Actually, a high density of polynomial can lead to exploding or vanishing behaviors.

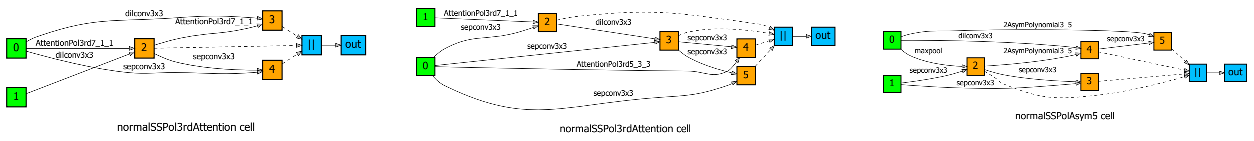


Figure 10: The diagrams are showing the best architectures found on each dataset using the polynomial search spaces and FairDARTS. The best ones found on SC09 are depicted on the left, the middle one corresponds to SC35 and the right one exhibits the best architecture learned on ESC50.

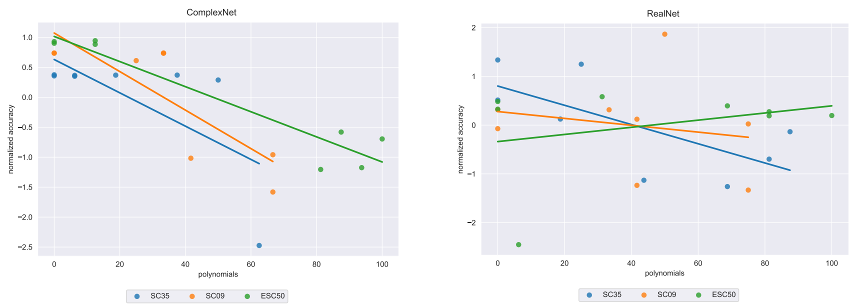


Figure 11: The relation between the percentage of polynomial operations in the architecture and the normalized accuracy is depicted. The regression plot on the left depicts this relation for networks using complex operations, while the right one depicts it for networks using real operations. The accuracy is standardized (substract the mean, divide by standard deviation) such that the relation can be exhibited independently of the dataset.

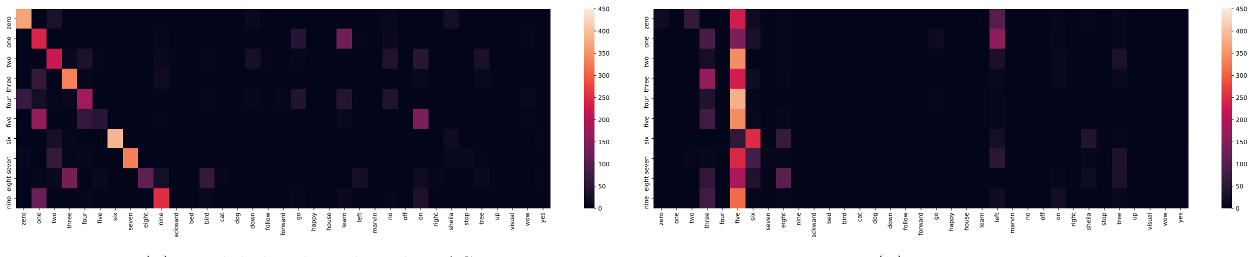
To avoid them, as depicted in figure 5, instance norm and batch norm blocks are introduced in combination with the hyperbolic tangent to clamp the values in [-1,1]. While such operations are working in the real field, it fails in the complex field. Finding a good operation/way to control the polynomial operations could then drastically reduce the risk of explosion/vanishing and then the risk of having degenerated architectures. Indeed, because when the architecture is not degenerated, polynomial search spaces are showing promising results and beats the unary search spaces.

Finally, we retain from those experiments that the risk of having a degenerated architecture is higher in the complex field when the density of polynomial operations is high. Based on figure 11, it seems that there exists non-zero optimal density for polynomial operations located between 10 and 40 percent, depending on the dataset. Also, when the architecture is not degenerated, the performances of polynomial are comparable with the ones obtained with unary operations. The better performances are mainly reached using real networks.

5.3 Transferability of the models learned

Beyond assessing the performances of our NAS methods, and the influence of pairwise operations, one can also assess the transferability of the architectures found. In other words, are the architectures found specific to the dataset they were trained on, or are they discovering a general structure transferable to other models ? To answer this question, the networks trained on SC35 and ESC50 were tested on AudioMNIST with and without a re-training stage. The results are reported in table 5.

Let's first analyze the performances without a re-training stage. The confusion matrix of a model with an architecture provided by ForceDARTS and the one of a ResNet18 (both trained on SC35),



(a) model developed with NAS (b) ResNet18

Figure 12: The heat maps above are depicting the confusion matrices of two models initially trained on SC35 and then transferred to AudioMNIST. The x-axis represents the class of SC35 (where the ten first ones are the spoken digits). The y-axis represents the classes of AudioMNIST, namely the spoken digits. On the left, the confusion matrix of a complex model whose architecture was found using ForceDARTS and **SS1** search space is exhibited, while on the right, the confusion matrix of a ResNet18 model trained on SC35 is exhibited.

Dataset	Model	Search space	Accuracy(%) without RT	Accuracy(%) with RT	Params(M)
SC35	ComplexNet	SS1	**38.58 ± 6.59**	94.37 ± 0.2	4.75
		SSPolAsym5	**35.41 ± 3.67**	**95.61 ± 0.29**	4.11
	RealNet	SS1	9.78 ± 2.07	93.9 ± 0.8	4.29
		SSPol3rdAttention	16.37 ± 5.01	**95.16 ± 0.2**	3.72
	ResNet18	_	13.21 ± 2.36	88.15 ± 0.1	11.12
ESC50	ComplexNet	SS1	_	**88.9 ± 0.49**	3.85
		SSPolAsym5	_	85.81 ± 0.45	4.18
	RealNet	SS1	_	86.3 ± 0.75	2.23
		SSPolAsym5	_	**88.49 ± 0.43**	3.16
	ResNet18	_	_	83.46 ± 0.12	11.12

Table 5: The performances of the models found using NAS methods on AudioMNIST are reported in the above table. The models selected are the ones that have obtained the best performances on SC35 and ESC50 according to tables 3 and 4. For SC35, since classes of AudioMNIST are included in SC35 (but data points are quite different: different sampling rate), it is relevant to record the accuracy without re-training the last layer: this corresponds to the **Accuracy without RT**. The column **Accuracy with RT** reports the performances after having re-trained the last layer on AudioMNIST. For each dataset, The top 2 performances are highlighted.

are exhibited on figure 12. While the performances of ResNet18 are poor (mainly predicting the class *five*), a diagonal stands out in the confusion matrix of the model developed with NAS method. This result is confirmed by table 5 where complex models from NAS exhibits an accuracy 20% higher than ResNet18 one. This statement is unfortunately not true for the real models that are showing similar performances to ResNet18 one. It then suggests that complex models are more likely to discover a general structure.

The performances reached after a re-training stage (last linear layer only) supports the previous observation. Regarding model trained on SC35, an accuracy of at least **94.37%** in the complex field and **93.9%** in the real field are reached against **88.15%** for the ResNet18 model. It actually confirms that the features generated by the convolution part of the architecture are not specific to a dataset but rather provides an embedding space where samples are easily separable. Regarding model trained on ESC50, an accuracy of at least **85.81%** in the complex field and **86.3%** in the real field are reached against **83.46%** for the ResNet18 model. What it is surprising with those results is that models trained on ESC50 were actually taking as input an audio spectrogram of size 56x146 while audio spectrogram of AudioMNIST are of size 32x32 and those models. Despite this difference, models are still providing reasonable performances.

6 Discussion

First, regarding ForceDARTS, it overcomes DARTS and shows comparable performances. This method could be improved fine-tuning its parameters, namely the weights associated with the impurity loss and the way of activating the loss. Indeed, the choice of those parameters were inspired from the ones used in FairDARTS paper [8]. Also, it could be interesting to monitor the performances of ForceDARTS on other datasets, especially image datasets.

Secondly, regarding the diversification of the search space, polynomial operations perform as good as unary operations if the architecture is not degenerated. Indeed, the risk of getting a degenerated architecture with polynomial operations is higher. To limit this risk, one could try to limit the magnitude of the variables involved in the product operation by increasing the weight decay only on parameters involved in polynomial candidate operations. In addition, adding a learnable parameter that multiplies the output of the product could control it.

7 Conclusion

The initial objective of this work was to explore new and existing methods in the context of NAS in complex field for audio recognition. A variant of DARTS [3], namely ForceDARTS was proposed and showed better performances than DARTS and comparable ones with FairDARTS [8]. On top of that, the search space was diversified introducing pairwise operations. This diversification showed similar results to the ones obtained with the original search space, namely **SS1**. Nevertheless, the risk of getting a degenerated architecture was higher, especially when the density of polynomial operations was large. Finally, the transferability of the models learned to use architectures found with NAS was evaluated. Compared to a ResNet18 model, the ones coming from NAS showed much better performances, suggesting that the architectures found are not dataset-specific.

8 Acknowledgment

I would like to warmly thank my supervisor, Grigoris Chrysos, that helped me throughout this project. He gave me many insights on which pathway I should choose. This project was a great opportunity to feel what is done and how things are done in the NAS field. In addition, this project allowed me to greatly enhance my programming skills as well as expand my knowledge. I would also like to thank professor Volkan Cehver that hosted me in his lab for this project.

References

[1] C. Liu, B. Zoph, M. Neumann, J. Shlens, W. Hua, J. Li, F.-F. Li, A. Yuille, J. Huang, and K. Murphy, "Progressive neural architecture search," vol. abs/1712.00559, 2017.

[2] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, OpenReview.net, 2017.

[3] H. Liu, K. Simonyan, and Y. Yang, "DARTS: differentiable architecture search," in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, OpenReview.net, 2019.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pp. 770–778, IEEE Computer Society, 2016.

[5] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA* (I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, eds.), pp. 5998–6008, 2017.

[7] S. Hershey, S. Chaudhuri, D. P. W. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, M. Slaney, R. J. Weiss, and K. W. Wilson, "CNN architectures for large-scale audio classification," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2017, New Orleans, LA, USA, March 5-9, 2017*, pp. 131–135, IEEE, 2017.

[8] X. Chu, T. Zhou, B. Zhang, and J. Li, "Fair darts: Eliminating unfair advantages in differentiable architecture search," 2019.

[9] C. Chiu, T. N. Sainath, Y. Wu, R. Prabhavalkar, P. Nguyen, Z. Chen, A. Kannan, R. J. Weiss, K. Rao, E. Gonina, N. Jaitly, B. Li, J. Chorowski, and M. Bacchiani, "State-of-the-art speech recognition with sequence-to-sequence models," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2018, Calgary, AB, Canada, April 15-20, 2018*, pp. 4774–4778, IEEE, 2018.

[10] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual* (H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, eds.), 2020.

[11] H. Lee, P. T. Pham, Y. Largman, and A. Y. Ng, "Unsupervised feature learning for audio classification using convolutional deep belief networks," in *Advances in Neural Information Processing Systems 22: 23rd Annual Conference on Neural Information Processing Systems 2009. Proceedings of a meeting held 7-10 December 2009, Vancouver, British Columbia, Canada* (Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, eds.), pp. 1096–1104, Curran Associates, Inc., 2009.

[12] Y. Gong, Y. Chung, and J. R. Glass, "AST: audio spectrogram transformer," in *Interspeech 2021, 22nd Annual Conference of the International Speech Communication Association, Brno, Czechia, 30 August - 3 September 2021* (H. Hermansky, H. Cernocký, L. Burget, L. Lamel, O. Scharenborg, and P. Motlícek, eds.), pp. 571–575, ISCA, 2021.

[13] A. Eronen, V. Peltonen, J. Tuomi, A. Klapuri, S. Fagerlund, T. Sorsa, G. Lorho, and J. Huopaniemi, "Audio-based context recognition," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 1, pp. 321–329, 2006.

[14] D. Stowell, D. Giannoulis, E. Benetos, M. Lagrange, and M. D. Plumbley, "Detection and classification of acoustic scenes and events," *IEEE Transactions on Multimedia*, vol. 17, no. 10, pp. 1733–1746, 2015.

[15] K. O'Shea and R. Nash, "An introduction to convolutional neural networks," 2015.

[16] P. Ren, Y. Xiao, X. Chang, P.-Y. Huang, Z. Li, X. Chen, and X. Wang, "A comprehensive survey of neural architecture search: Challenges and solutions," 2020.

[17] C. Liu, L. Chen, F. Schroff, H. Adam, W. Hua, A. L. Yuille, and F. Li, "Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pp. 82–92, Computer Vision Foundation / IEEE, 2019.

[18] G. Bender, P. Kindermans, B. Zoph, V. Vasudevan, and Q. V. Le, "Understanding and simplifying one-shot architecture search," in *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018* (J. G. Dy and A. Krause, eds.), vol. 80 of *Proceedings of Machine Learning Research*, pp. 549–558, PMLR, 2018.

[19] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," in *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pp. 4780–4789, AAAI Press, 2019.

[20] A. Zela, T. Elsken, T. Saikia, Y. Marrakchi, T. Brox, and F. Hutter, "Understanding and robustifying differentiable architecture search," in *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, OpenReview.net, 2020.

[21] X. Chen, L. Xie, J. Wu, and Q. Tian, "Progressive darts: Bridging the optimization gap for nas in the wild," 2019.

[22] Y. Chen, J. Hsu, C. Lee, and H. Lee, "DARTS-ASR: differentiable architecture search for multilingual speech recognition and adaptation," in *Interspeech 2020, 21st Annual Conference of the International Speech Communication Association, Virtual Event, Shanghai, China, 25-29 October 2020* (H. Meng, B. Xu, and T. F. Zheng, eds.), pp. 1803–1807, ISCA, 2020.

[23] S. Ding, T. Chen, X. Gong, W. Zha, and Z. Wang, "Autospeech: Neural architecture search for speaker recognition," in *Interspeech 2020, 21st Annual Conference of the International Speech Communication Association, Virtual Event, Shanghai, China, 25-29 October 2020* (H. Meng, B. Xu, and T. F. Zheng, eds.), pp. 916–920, ISCA, 2020.

[24] T. Mo, Y. Yu, M. Salameh, D. Niu, and S. Jui, "Neural architecture search for keyword spotting," in *Interspeech 2020, 21st Annual Conference of the International Speech Communication Association, Virtual Event, Shanghai, China, 25-29 October 2020* (H. Meng, B. Xu, and T. F. Zheng, eds.), pp. 1982–1986, ISCA, 2020.

[25] G. G. Chrysos, S. Moschoglou, G. Bouritsas, Y. Panagakis, J. Deng, and S. Zafeiriou, "$\pi-$nets: Deep polynomial neural networks," 2020.

[26] C. Trabelsi, O. Bilaniuk, Y. Zhang, D. Serdyuk, S. Subramanian, J. F. Santos, S. Mehri, N. Rostamzadeh, Y. Bengio, and C. J. Pal, "Deep complex networks," in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, OpenReview.net, 2018.

[27] S. Becker, M. Ackermann, S. Lapuschkin, K.-R. Müller, and W. Samek, "Interpreting and explaining deep neural networks for classification of audio signals," *CoRR*, vol. abs/1807.03418, 2018.

[28] P. Warden, "Speech commands: A dataset for limited-vocabulary speech recognition," 2018.

[29] K. J. Piczak, "ESC: Dataset for Environmental Sound Classification," in *Proceedings of the 23rd Annual ACM Conference on Multimedia*, pp. 1015–1018, ACM Press.

[30] H. Zhang, M. Cissé, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, OpenReview.net, 2018.

[31] Microsoft, "Neural Network Intelligence," 2021.

[32] J. Kileel, M. Trager, and J. Bruna, "On the expressive power of deep polynomial neural networks," in *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada* (H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, eds.), pp. 10310–10319, 2019.