ResMLP: Feedforward Networks for Image Classification With Data-Efficient Training

Hugo Touvron, Piotr Bojanowski, Mathilde Caron, Matthieu Cord, Alaaeldin El-Nouby, Edouard Grave, Gautier Izacard, Armand Joulin, Gabriel Synnaeve, Jakob Verbeek, and Hervé Jégou

Abstract—We present ResMLP, an architecture built entirely upon multi-layer perceptrons for image classification. It is a simple residual network that alternates (i) a linear layer in which image patches interact, independently and identically across channels, and (ii) a two-layer feed-forward network in which channels interact independently per patch. When trained with a modern training strategy using heavy data-augmentation and optionally distillation, it attains surprisingly good accuracy/complexity trade-offs on ImageNet. We also train ResMLP models in a self-supervised setup, to further remove priors from employing a labelled dataset. Finally, by adapting our model to machine translation we achieve surprisingly good results. We share pre-trained models and our code based on the Timm library.

Index Terms—Multi-layer perceptron, computer-vision, NLP

1 Introduction

Recently, the transformer architecture [1], adapted from its original use in natural language processing with only minor changes, has achieved performance competitive with the state of the art on ImageNet-1k [2] when pre-trained with a sufficiently large amount of data [3]. Retrospectively, this achievement is another step towards learning visual features with less priors: Convolutional Neural Networks (CNN) had replaced the hand-designed choices from hard-wired features with flexible and trainable architectures. Vision transformers further removes several hard decisions encoded in the convolutional architectures, namely the translation invariance and local connectivity.

This evolution toward less hard-coded prior in the architecture has been fueled by better training schemes [3], [4], and, in this paper, we push this trend further by showing that a purely multi-layer perceptron (MLP) based architecture, called Residual Multi-Layer Perceptrons (ResMLP), is competitive on image classification. ResMLP is designed to be simple and encoding little prior about images: it takes image patches as input, projects them with a linear layer, and sequentially updates their representations with two residual operations: (i) a *cross-patch* linear layer applied to all channels independently; and (ii) an *cross-channel* single-layer MLP applied independently to all patches. At the end of the network, the patch representations are average pooled, and fed to a linear classifier. We outline ResMLP in Fig. 1 and detail it further in Section 2.

- *Hugo Touvron is with Facebook AI Research, 75004 Paris, France, and also with Sorbonne University, 75006 Paris, France. E-mail: htouvron@fb.com.*
- *Piotr Bojanowski, Mathilde Caron, Alaaeldin El-Nouby, Edouard Grave, Gautier Izacard, Armand Joulin, Gabriel Synnaeve, Jakob Verbeek, and Hervé Jégou are with Facebook AI Research, 75004 Paris, France. E-mail: {bojanowski, mathilde, aelnouby, egrave, gizacard, ajoulin, gab, jjverbeek, rvj}@fb.com.*
- *Matthieu Cord is with Sorbonne University, 75006 Paris, France. E-mail: matthieu.cord@sorbonne-universite.fr.*

Manuscript received 19 November 2021; revised 20 May 2022; accepted 2 August 2022. Date of publication 12 September 2022; date of current version 6 March 2023. (Corresponding author: Hugo Touvron.) Recommended for acceptance by K. Fragkiadaki. This article has supplementary downloadable material available at https://doi.org/10.1109/TPAMI.2022.3206148, provided by the authors. Digital Object Identifier no. 10.1109/TPAMI.2022.3206148

The ResMLP architecture is strongly inspired by the vision transformers (ViT) [3], yet it is much simpler in several ways: we replace the self-attention sublayer by a linear layer, resulting in an architecture with only linear layers and GELU non-linearity [5]. We observe that the training of ResMLP is more stable than ViTs when using the same training scheme as in DeiT [4] and CaiT [6], removing the need for batch-specific or cross-channel normalizations such as BatchNorm, GroupNorm or LayerNorm. We speculate that this stability comes from replacing self-attention with linear layers. Finally, another advantage of using a linear layer is that we can still visualize the interactions between patch embeddings, revealing filters that are similar to convolutions on the lower layers, and longer range in the last layers.

We further investigate if our purely MLP based architecture could benefit to other domains beyond images, and particularly, with more complex output spaces. In particular, we adapt our MLP based architecture to take inputs with variable length, and show its potential on the problem of Machine Translation. To do so, we develop a sequence-to-sequence (seq2seq) version of ResMLP, where both encoder and decoders are based on ResMLP with across-attention between the encoder and decoder [7]. This model is similar to the original seq2seq Transformer with ResMLP layers instead of Transformer layers [1]. Despite not being originally designed for this task, we observe that ResMLP is competitive with Transformers on the challenging WMT benchmarks.

In summary, in this paper, the main contributions that we consider in our paper are the following:

- We propose a drastically simplified design for neural network, namely ResMLP, whose main difference compared to a MLP is the residual design borrowed from earlier convolutional architectures.
- Albeit simple, ResMLP reaches surprisingly decent accuracy/complexity trade-offs when trained on ImageNet-1k only
- these models benefit significantly from distillation methods [4] and are easily combined with modern self-supervised learning methods based on data augmentation, such as DINO [8];
- A seq2seq ResMLP achieves competitive performances compared to a seq2seq Transformers on the WMT benchmark for Machine Translation.

2 Method

In this section, we describe our architecture, ResMLP, as depicted in Fig. 1. ResMLP is inspired by ViT and this section focuses on the changes made to ViT that lead to a purely MLP based model. We refer to Dosovitskiy et al. [3] for more details about ViT.

The Overall ResMLP Architecture. Our model, denoted by ResMLP, takes a grid of $N \times N$ non-overlapping patches as input, where the patch size is typically equal to 16×16. The patches are then independently passed through a linear layer to form a set of N^2 d-dimensional embeddings.

The resulting set of N^2 embeddings are fed to a sequence of *Residual Multi-Layer Perceptron* layers to produce a set of N^2 d-dimensional output embeddings. These output embeddings are then averaged ("average-pooling") as a d-dimension vector to represent the image, which is fed to a linear classifier to predict the label associated with the image. Training uses the cross-entropy loss.

The Residual Multi-Perceptron Layer. Our network is a sequence of layers that all have the same structure: a linear sublayer applied across patches followed by a feedforward sublayer applied across channels. Similar to the Transformer layer, each sublayer is

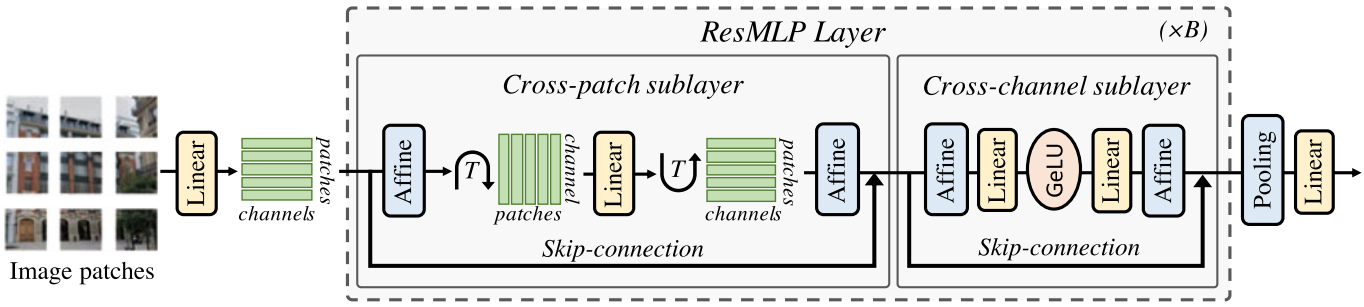

Fig. 1. *The ResMLP architecture.* After linearly projecting the image patches into high dimensional embeddings, ResMLP sequentially processes them with (1) a cross-patch linear sublayer; (2) a cross-channel two-layer MLP. The MLP is the same as the FCN sublayer of a Transformer. Each sublayer has a residual connection and two Affine element-wise transformations.

paralleled with a skip-connection [9]. The absence of self-attention layers makes the training more stable, allowing us to replace the Layer Normalization [10] by a simpler Affine transformation

$$\texttt{Aff}_{\boldsymbol{\alpha},\boldsymbol{\beta}}(\mathbf{x}) = \texttt{Diag}(\boldsymbol{\alpha})\mathbf{x} + \boldsymbol{\beta}, \tag{1}$$

where $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are learnable weight vectors. This operation only rescales and shifts the input element-wise. This operation has several advantages over other normalization operations: first, as opposed to Layer Normalization, it has no cost at inference time, since it can be absorbed in the adjacent linear layer. Second, as opposed to BatchNorm [11] and Layer Normalization, the `Aff` operator does not depend on batch statistics. The closer operator to `Aff` is the LayerScale introduced by Touvron et al. [6], with an additional bias term. For convenience, we denote by $\texttt{Aff}(\mathbf{X})$ the Affine operation applied independently to each column of the matrix $\mathbf{X}$.

We apply the `Aff` operator at the beginning ("pre-normalization") and end ("post-normalization") of each residual block. As a pre-normalization, `Aff` replaces LayerNorm without using channel-wise statistics. Here, we initialize $\boldsymbol{\alpha} = \mathbf{1}$, and $\boldsymbol{\beta} = \mathbf{0}$. As a post-normalization, `Aff` is similar to LayerScale and we initialize $\boldsymbol{\alpha}$ with the same small value as in [6].

Overall, our Multi-layer perceptron takes a set of N^2 d-dimensional input features stacked in a $d \times N^2$ matrix $\mathbf{X}$, and outputs a set of N^2 d-dimension output features, stacked in a matrix $\mathbf{Y}$ with the following set of transformations:

$$\mathbf{Z} = \mathbf{X} + \texttt{Aff}\Big((\mathbf{A} \cdot \texttt{Aff}\big(\mathbf{X}\big)^\top\Big)^\top\Big), \tag{2}$$

$$\mathbf{Y} = \mathbf{Z} + \texttt{Aff}(\mathbf{C} \cdot \texttt{GELU}(\mathbf{B} \cdot \texttt{Aff}(\mathbf{Z}))), \tag{3}$$

where $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ are the main learnable weight matrices of the layer. Note that Eq. (3) is the same as the feedforward sublayer of a Transformer with the `ReLU` non-linearity replaced by a `GELU` function [5]. The dimensions of the parameter matrix $\mathbf{A}$ are $N^2 \times N^2$, i.e., this "cross-patch" sublayer exchanges information between patches, while the "cross-channel" feedforward sublayer works per location. Similar to a Transformer, the intermediate activation matrix $\mathbf{Z}$ has the same dimensions as the input and output matrices, $\mathbf{X}$ and $\mathbf{Y}$. Finally, the weight matrices $\mathbf{B}$ and $\mathbf{C}$ have the same dimensions as in a Transformer layer, which are $4d \times d$ and $d \times 4d$, respectively.

Differences With the Vision Transformer Architecture. Our architecture is closely related to the ViT model [3]. However, ResMLP departs from ViT with several simplifications:

- *no self-attention blocks*: it is replaced by a linear layer with no non-linearity,
- *no positional embedding*: the linear layer implicitly encodes information about patch positions,
- *no extra "class" token*: we simply use average pooling on the patch embeddings,
- *no normalization based on batch statistics*: we use a learnable affine operator.

Class-MLP as an Alternative to Average Pooling. We propose an adaptation of the class-attention token introduced in CaiT [6]. In CaiT, this consists of two layers that have the same structure as the transformer, but in which only the class token is updated based on the frozen patch embeddings. We translate this method to our architecture, except that, after aggregating the patches with a linear layer, we replace the attention-based interaction between the class and patch embeddings by simple linear layers, still keeping the patch embeddings frozen. This increases the performance, at the expense of adding some parameters and computational cost. We refer to this pooling variant as "class-MLP", since the purpose of these few layers is to replace average pooling.

Sequence-to-Sequence ResMLP. Similar to Transformer, the ResMLP architecture can be applied to sequence-to-sequence tasks. First, we follow the general encoder-decoder architecture from Vaswani et al. [1], where we replace the self-attention sublayers by the residual multi-perceptron layer. In the decoder, we keep the cross-attention sublayers, which attend to the output of the encoder. In the decoder, we adapt the linear sublayers to the task of language modeling by constraining the matrix $\mathbf{A}$ to be triangular, in order to prevent a given token representation to access tokens from the future. Finally, the main technical difficulty from using linear sublayers in a sequence-to-sequence model is to deal with variable sequence lengths. However, we observe that simply padding with zeros and extracting the submatrix $\mathbf{A}$ corresponding to the longest sequence in a batch, works well in practice.

3 Experiments

In this section, we present experimental results for the ResMLP architecture on image classification and machine translation. We also study the impact of the different components of ResMLP in ablation studies. We consider three training paradigms for images:

- *Supervised learning:* We train ResMLP from labeled images with a softmax classifier and cross-entropy loss. This paradigm is the main focus of our work.
- *Self-supervised learning:* We train the ResMLP with the DINO method of Caron et al. [8] that trains a network without labels by distilling knowledge from previous instances of the same network.
- *Knowledge distillation:* We employ the knowledge distillation procedure proposed by Touvron et al. [4] to guide the supervised training of ResMLP with a convnet.

3.1 Experimental setting

Datasets. We train our models on the ImageNet-1k dataset [2], that contains 1.2M images evenly spread over 1,000 object categories. In

TABLE 1
Comparison Between Architectures on ImageNet Classification

	Arch.	#params ($\times 10^6$)	throughput (im/s)	FLOPS ($\times 10^9$)	Peak Mem (MB)	Top-1 Acc.
State of the art	CaiT-M48↑ 448ϒ [6]	356	5.4	329.6	5477.8	**86.5**
	NfNet-F6 SAM [16]	438	16.0	377.3	5519.3	**86.5**
Convolutional networks	EfficientNet-B3 [17]	12	661.8	1.8	1174.0	81.1
	EfficientNet-B4 [17]	19	349.4	4.2	1898.9	82.6
	EfficientNet-B5 [17]	30	169.1	9.9	2734.9	83.3
	RegNetY-4GF [18]	21	861.0	4.0	568.4	80.0
	RegNetY-8GF [18]	39	534.4	8.0	841.6	81.7
	RegNetY-16GF [18]	84	334.7	16.0	1329.6	82.9
Transformer networks	DeiT-S [4]	22	940.4	4.6	217.2	79.8
	DeiT-B [4]	86	292.3	17.5	573.7	81.8
	CaiT-XS24 [6]	27	447.6	5.4	245.5	81.8
Feedforward networks	ResMLP-S12	15	1415.1	3.0	179.5	76.6
	ResMLP-S24	30	715.4	6.0	235.3	79.4
	ResMLP-B24	116	231.3	23.0	663.0	81.0

We compare different architectures based on convolutional networks, Transformers and feedforward networks with comparable FLOPs and number of parameters. We report Top-1 accuracy on the validation set of ImageNet-1k with different measure of complexity: throughput, FLOPs, number of parameters and peak memory usage. All the models use 224×224 images as input. By default the Transformers and feedforward networks uses 14×14 patches of size 16×16, see Table 3 for the detailed specification of our main models. The throughput is measured on a single V100-32GB GPU with batch size fixed to 32. For reference, we include the state of the art with ImageNet training only.

the absence of an available test set for this benchmark, we follow the standard practice in the community by reporting performance on the validation set. This is not ideal since the validation set was originally designed to select hyper-parameters. Comparing methods on this set may not be conclusive enough because an improvement in performance may not be caused by better modeling, but by a better selection of hyper-parameters. To mitigate this risk, we report additional results in transfer learning and on two alternative versions of ImageNet that have been built to have distinct validation and test sets, namely the ImageNet-real [12] and ImageNet-v2 [13] datasets. We also report a few data-points when training on ImageNet-21k. Our hyper-parameters are mostly adopted from Touvron et al. [4], [6].

Hyper-Parameter Settings. In the case of supervised learning, we train our network with the Lamb optimizer [14] with a learning rate of 5×10^{-3} and weight decay 0.2. We initialize the LayerScale parameters as a function of the depth by following CaiT [6]. The rest of the hyper-parameters follow the default setting used in DeiT [4]. For the knowledge distillation paradigm, we use the same RegNety-16GF [15] as in DeiT with the same training schedule. The majority of our models take two days to train on eight V100-32GB GPUs.

3.2 Main Results

In this section, we compare ResMLP with architectures based on convolutions or self-attentions with comparable size and throughput on ImageNet.

Supervised Setting. In Table 1, we compare ResMLP with different convolutional and Transformer architectures. For completeness, we also report the best-published numbers obtained with a model trained on ImageNet alone. While the trade-off between accuracy, FLOPs, and throughput for ResMLP is not as good as convolutional networks or Transformers, their strong accuracy still suggests that the structural constraints imposed by the layer design do not have a drastic influence on performance, especially when training with enough data and recent training schemes.

Self-Supervised Setting. We pre-train ResMLP-S12 using the self-supervised method called DINO [8] during 300 epochs. We report our results in Table 2. The trend is similar to the supervised setting: the accuracy obtained with ResMLP is lower than ViT. Nevertheless, the performance is surprisingly high for a pure MLP architecture and competitive with Convnet in k-NN evaluation. Additionally, we also fine-tune a model pre-trained with self-supervision on ImageNet using the ground-truth labels. Pre-training substantially improves performance compared to a ResMLP-S24 solely trained with labels, achieving 79.9% top-1 accuracy on ImageNet-val (+0.5%).

Knowledge Distillation Setting. We study our model when training with the knowledge distillation approach of Touvron et al. [4]. In their work, the authors show the impact of training a ViT model by distilling it from a RegNet. In this experiment, we explore if ResMLP also benefits from this procedure and summarize our results in Table 3 (Blocks "Baseline models" and "Training"). We observe that similar to DeiT models, ResMLP greatly benefits from distilling from a convnet. This result concurs with the observations made by d'Ascoli et al. [19], who used convnets to initialize feedforward networks. Even though our setting differs from theirs in scale, the problem of overfitting for feedforward networks is still present on ImageNet. The additional regularization obtained from the distillation is a possible explanation for this improvement.

3.3 Visualization & Analysis of the Linear Interaction Between Patches

In Fig. 2, we show in the form of squared images, the rows of the weight matrix from cross-patch sublayers at different depths of a ResMLP-S24 model. The early layers show convolution-like patterns: the weights resemble shifted versions of each other and have

TABLE 2
Self-Supervised Learning With DINO [8]

Models	Params. ($\times 10^6$)	FLOPS ($\times 10^9$)	ImNet-val top-1 acc. Linear	ImNet-val top-1 acc. k-NN
ResNet-50	25	4.1	75.3	67.5
ViT-S/16	22	4.6	77.0	74.5
ViT-S/8	22	22.4	**79.7**	**78.3**
ViT-B/16	87	17.5	78.2	76.1
ResMLP-S12	15	3.0	67.5	62.6
ResMLP-S24	30	6.0	72.8	69.4

Classification accuracy on ImageNet-1k val. ResMLPs evaluated with linear and k-NN evaluation are on par with convnets but inferior to ViTs.

TABLE 3
Ablation. Our Default Configurations are Presented in the Three First Rows

Ablation	Model	Patch size	Params $\times 10^6$	FLOPs $\times 10^9$	Variant	top-1 acc. on ImageNet		
						val	real [12]	v2 [13]
Baseline models	ResMLP-S12	16	15.4	3.0	12 layers, working dimension 384	76.6	83.3	64.4
	ResMLP-S24	16	30.0	6.0	24 layers, working dimension 384	79.4	85.3	67.9
	ResMLP-B24	16	115.7	23.0	24 layers, working dimension 768	81.0	86.1	69.0
Normalization	ResMLP-S12	16	15.4	3.0	`Aff` → Layernorm	77.7	84.1	65.7
Pooling	ResMLP-S12	16	17.7	3.0	average pooling → Class-MLP	77.5	84.0	66.1
Patch communication	ResMLP-S12	16	14.9	2.8	linear → none	56.5	63.4	43.1
	ResMLP-S12	16	18.6	4.3	linear → MLP	77.3	84.0	65.7
	ResMLP-S12	16	30.8	6.0	linear → conv 3x3	77.3	84.4	65.7
	ResMLP-S12	16	14.9	2.8	linear → conv 3x3 depth-wise	76.3	83.4	64.6
	ResMLP-S12	16	16.7	3.2	linear → conv 3x3 depth-separable	77.0	84.0	65.5
Patch size	ResMLP-S12/14	14	15.6	4.0	patch size 16×16 → 14×14	76.9	83.7	65.0
	ResMLP-S12/8	8	22.1	14.0	patch size 16×16 → 8×8	79.1	85.2	67.2
	ResMLP-B24/8	8	129.1	100.2	patch size 16×16 → 8×8	81.0	85.7	68.6
Training	ResMLP-S12	16	15.4	3.0	old-fashioned (90 epochs)	69.2	76.0	56.1
	ResMLP-S12	16	15.4	3.0	pre-trained SSL (DINO)	76.5	83.6	64.5
	ResMLP-S12	16	15.4	3.0	distillation	77.8	84.6	66.0
	ResMLP-S24	16	30.0	6.0	pre-trained SSL (DINO)	79.9	85.9	68.6
	ResMLP-S24	16	30.0	6.0	distillation	80.8	86.6	69.8
	ResMLP-B24/8	8	129.1	100.2	distillation	83.6	88.4	73.4
	ResMLP-B24/8	8	129.1	100.2	pre-trained ImageNet-21k (60 epochs)	**84.4**	**88.9**	**74.2**

By default we train during 400 epochs. The "old-fashioned" is similar to what was employed for ResNet [9]: SGD, 90-epochs waterfall schedule, same augmentations up to variations due to library used.

local support. Interestingly, in many layers, the support also extends along both axes; see layer 7. The last 7 layers of the network are different: they consist of a spike for the patch itself and a diffuse response across other patches with different magnitude; see layer 20.

Measuring Sparsity of the Weights. The visualizations described above suggest that the linear communication layers are sparse. We analyze this quantitatively in more detail in Fig. 3. We measure the sparsity of the matrix **A**, and compare it to the sparsity of **B** and **C** from the per-patch MLP. Since there are no exact zeros, we measure the rate of components whose absolute value is lower than 5% of the maximum value. Note, discarding the small values is analogous to the case where we normalize the matrix by its maximum and use a finite-precision representation of weights. For instance, with a 4-bits representation of weight, one would typically round to zero all weights whose absolute value is below 6.25% of the maximum value.

The measurements in Fig. 3 show that all three matrices are sparse, with the layers implementing the patch communication being significantly more so. This suggests that they may be compatible with parameter pruning, or better, with modern quantization techniques that induce sparsity at training time, such as Quant-Noise [20] and DiffQ [21]. The sparsity structure, in particular in earlier layers, see Fig. 2, hints that we could implement the patch interaction linear layer with a convolution. We provide some results for convolutional variants in our ablation study. Further research on network compression is beyond the scope of this paper, yet we believe it worth investigating in the future.

Communication Across Patches. If we remove the linear interaction layer (linear → none), we obtain substantially lower accuracy (-20% top-1 acc.) for a "bag-of-patches" approach. We have tried several alternatives for the cross-patch sublayer, which are presented in Table 3 (block "patch communication"). Amongst them, using the same MLP structure as for patch processing (linear → MLP), which we analyze in more details in the Appendix, available in the online supplemental material. The simpler choice of a single linear square layer led to a better accuracy/performance trade-off – considering that the MLP variant requires compute halfway

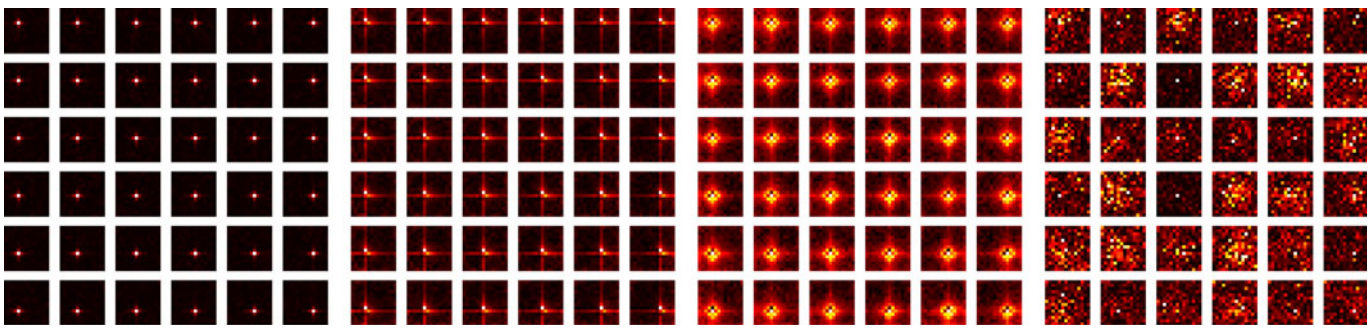

Fig. 2. *Visualisation of the linear layers in ResMLP-S24.* For each layer we visualise the rows of the matrix $\mathbf{A}$ as a set of 14×14 pixel images, for sake of space we only show the rows corresponding to the 6×6 central patches. We observe patterns in the linear layers that share similarities with convolutions. In Appendix B, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPAMI.2022.3206148, we provide comparable visualizations for all layers of a ResMLP-S12 model.

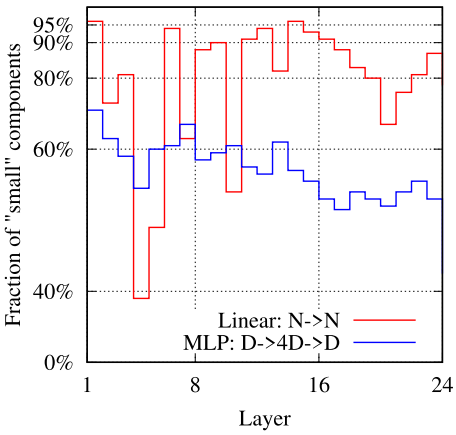

Fig. 3. *Sparsity of linear interaction layers.* For each layer (linear and MLP), we show the rate of components whose absolute value is lower than 5% of the maximum. Linear interaction layers are sparser than the matrices involved in the per-patch MLP.

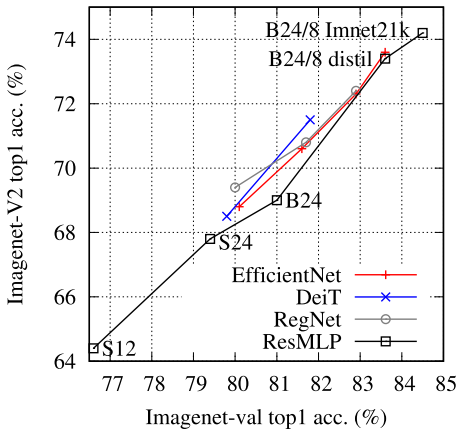

Fig. 4. *Top-1 accuracy on ImageNet-V2 versus ImageNet-val.* ResMLPs tend to overfit slightly more under identical training method. This is partially alleviated with by introducing more regularization (more data or distillation, see e.g., ResMLP-B24/8-distil).

between ResMLP-S12 and ResMLP-S24 – and requires fewer parameters than a residual MLP block.

The visualization in Fig. 2 indicates that many linear interaction layers look like convolutions, the kernels being increasingly larger closer to the output layer. Hence in our ablation, we replaced the linear layer with different types of 3×3 convolutions. The depth-wise convolution does not implement interaction across channels – as our linear patch communication layer – and yields similar performance with a comparable number of parameters and FLOPs. While full 3×3 convolutions yield best results, they require roughly double the number of parameters and FLOPs. Interestingly, the depth-separable convolutions combine accuracy close to that of full 3×3 convolutions with a number of parameters and FLOPs comparable to our linear layer. This suggests that convolutions on low-resolution feature maps at all layers is an interesting alternative to the common pyramidal design of convnets, where early layers operate at higher resolution and smaller feature dimension.

3.4 Ablation Studies

Table 3 reports the ablation study of our base network and a summary of our preliminary exploratory studies. We discuss the ablation below and give more detail about early experiments in Appendix A, available in the online supplemental material.

Control of Overfitting. Since MLPs are subject to overfitting, we show in Fig. 4a control experiment to probe for problems with generalization. We explicitly analyze the differential of performance between the ImageNet-val and the distinct ImageNet-V2 test set. The relative offsets between curves reflect to which extent models are overfitted to ImageNet-val w.r.t. hyper-parameter selection. The degree of overfitting of our MLP-based model is overall neutral or slightly higher to that of other transformer-based architectures or convnets with same training procedure.

Normalization & Activation. Our network configuration does not contain any batch normalizations. Instead, we use the affine per-channel transform `Aff`. This is akin to Layer Normalization [10], typically used in transformers, except that we avoid to collect any sort of statistics, since we do no need it it for convergence. In preliminary experiments with pre-norm and post-norm [22], we observed that both choices converged. Pre-normalization in conjunction with Batch Normalization could provide an accuracy gain in some cases, see Appendix A, available in the online supplemental material.

We choose to use a GELU [5] function. In Appendix A, available in the online supplemental material, we also analyze the activation function: ReLU [23] also gives a good performance, but it was a bit more unstable in some settings. We did not manage to get good results with SiLU [5] and HardSwish [24].

Pooling. Replacing average pooling with Class-MLP, see Section 2, brings a significant gain for a negligible computational cost. We do not include it by default to keep our models more simple.

Patch Size. Smaller patches significantly increase the performance, but also increase the number of flops (see Block "Patch size" in Table 3). Smaller patches benefit more to larger models, but only with an improved optimization scheme involving more regularization (distillation) or more data.

Training. Consider the Block "Training' in Table 3. ResMLP significantly benefits from modern training procedures such as those used in DeiT. For instance, the DeiT training procedure improves the performance of ResMLP-S12 by 7.4% compared to the training

TABLE 4
Evaluation on Transfer Learning

Architecture	FLOPs	Res.	CIFAR_{10}	CIFAR_{100}	Flowers102	Cars	iNat_{18}	iNat_{19}
EfficientNet-B7 [17]	37.0B	600	98.9	**91.7**	**98.8**	**94.7**	_	_
ViT-B/16 [3]	55.5B	384	98.1	87.1	89.5	_	_	_
ViT-L/16 [3]	190.7B	384	97.9	86.4	89.7	_	_	_
Deit-B/16 [4]	17.5B	224	**99.1**	90.8	98.4	92.1	**73.2**	**77.7**
ResNet50 [31]	4.1B	224	_	_	96.2	90.0	68.4	73.7
Grafit/ResNet50 [31]	4.1B	224	_	_	97.6	92.7	68.5	74.6
ResMLP-S12	3.0B	224	98.1	87.0	97.4	84.6	60.2	71.0
ResMLP-S24	6.0B	224	98.7	89.5	97.9	89.5	64.3	72.5

Classification accuracy (top-1) of models trained on ImageNet-1k for transfer to datasets covering different domains. The ResMLP architecture takes 224×224 *images during training and transfer, while ViTs and EfficientNet-B7 work with higher resolutions, see "Res." column.*

TABLE 5
Machine Translation on WMT 2014 Translation Tasks

Models	GNMT [33]	ConvS2S [34]	Transf. (base) [1]	ResMLP-6	ResMLP-12
En-De	24.6	25.2	**27.3**	26.4	26.8
En-Fr	39.9	40.5	38.1	40.3	**40.6**

We report tokenized BLEU on newstest2014.

TABLE 6
Semantic Segmentation Results on ADE20K Dataset With UperNet and $\times 3$ Settings

Model	DeiT-S	DeiT-B	ResMLP-12	ResMLP-24	ResMLP-36	ResMLP-B24
mIoU (%)	40.5	42.3	35.9	39.1	39.1	**42.5**

All architectures are train with crop of size 224×224.

employed for ResNet [9].[1] This is in line with recent work pointing out the importance of the training strategy over the model choice [15], [26]. Pre-training on more data and distillation also improve the performance of ResMLP, especially for the bigger models, e.g., distillation improves the accuracy of ResMLP-B24/8 by 2.6%.

Other Analysis. In our early exploration, we evaluated several alternative design choices. As in transformers, we could use positional embeddings mixed with the input patches. In our experiments we did not see any benefit from using these features, see Appendix A, available in the online supplemental material. This observation suggests that our cross-patch sublayer provides sufficient spatial communication, and referencing absolute positions obviates the need for any form of positional encoding.

Transfer Learning. We evaluate the quality of features obtained from a ResMLP architecture when transferring them to other domains. The goal is to assess if the features generated from a feedforward network are more prone to overfitting on the training data distribution. We adopt the typical setting where we pre-train a model on ImageNet-1k and fine-tune it on the training set associated with a specific domain. We report the performance with different architectures on various image benchmarks in Table 4, namely CIFAR-10 and CIFAR-100 [27], Flowers-102 [28], Stanford Cars [29] and iNaturalist [30]. We refer the reader to the corresponding references for a more detailed description of the datasets.

We observe that the performance of our ResMLP is competitive with the existing architectures, showing that pretraining feedforward models with enough data and regularization via data augmentation greatly reduces their tendency to overfit on the original distribution. Interestingly, this regularization also prevents them from overfitting on the training set of smaller dataset during the fine-tuning stage.

3.5 Machine Translation

We also evaluate the ResMLP transpose-mechanism to replace the self-attention in the encoder and decoder of a neural machine translation system. We train models on the WMT 2014 English-German and English-French tasks, following the setup from Ott et al. [32]. We consider models of dimension 512, with a hidden MLP size of 2,048, and with 6 or 12 layers. Note that the current state of the art employs much larger models: our 6-layer model is more comparable to the base transformer model from Vaswani et al. [1], which serves as a baseline, along with pre-transformer architectures such as recurrent and convolutional neural networks. We use Adagrad with learning rate 0.2, 32k steps of linear warmup, label smoothing 0.1, dropout rate 0.15 for En-De and 0.1 for En-Fr. We initialize the LayerScale parameter to 0.2. We generate translations with the beam search algorithm, with a beam of size 4. As shown in Table 5, the results are at least on par with the compared architectures. We do not replace the cross attention layers with an MLP because the input-adaptive aspect seems essential for Seq2Seq tasks.

1. Interestingly, if trained with this "old-fashion" setting, ResMLP-S12 outperforms AlexNet [25] by a margin.

3.6 Semantic Segmentation

We perform semantic segmentation experiments on the ADE20k [35] datasets with ResMLP models pre-trained on ImageNet. We adopt the classical UperNet [36] setting with the $\times 3$ schedule [37], [38]. We finetune for the semantic segmentation task at resolution 224×224 and evaluate the network with the usual resolution by adopting a sliding window. We report results in Table 6 and compare ResMLP with DeiT with the same setting. In that case ResMLP is used as backbone in order to extract features before the UperNet part. At training time we resize images at resolution 896×224 and apply RandomResizeCrop in order to have an image of size 224×224. ResMLP obtains interesting results, but note that vision transformers remain better. One of the main limitations is the fixed spatial linear layer which makes it more difficult to adapt the ResMLP architecture at different resolutions. Indeed, the size of the linear layer applied to the spatial dimension is fixed and can only be used with a certain image size. It is possible to interpolate the weights to adapt to other resolutions but this does not allow a very good adaptability.

4 Related Work

We review the research on applying Fully Connected Network (FCN) for computer vision problems as well as other architectures that shares common modules with our model.

Fully-Connected Network for Images. Many studies have shown that FCNs are competitive with convnets for the tasks of digit recognition [39], [40], keyword spotting [41] and handwritting recognition [42]. Several works [43], [44], [45] have questioned if FCNs are also competitive on natural image datasets, such as CIFAR-10 [27]. More recently, d'Ascoli et al. [19] have shown that a FCN initialized with the weights of a pretrained convnet achieves performance that are superior than the original convnet. Neyshabur [46] further extend this line of work by achieving competitive performance by training an FCN from scratch but with a regularizer that constrains the models to be close to a convnet. These studies have been conducted on small scale datasets with the purpose of studying the impact of architectures on generalization in terms of sample complexity [47] and energy landscape [48]. In our work, we show that, in the larger scale setting of ImageNet, FCNs can attain surprising accuracy without any constraint or initialization inspired by convnets.

Finally, the application of FCN networks in computer vision have also emerged in the study of the properties of networks with infinite width [49], or for inverse scattering problems [50]. More interestingly, the Tensorizing Network [51] is an approximation of very large FCN that shares similarity with our model, in that they intend to remove prior by approximating even more general tensor operations, i.e., not arbitrarily marginalized along some predefined sharing dimensions. However, their method is designed to compress the MLP layers of a standard convnets.

Other Architectures With Similar Components. Our FCN architecture shares several components with other architectures, such as convnets [25], [52] or transformers [1]. A fully connected layer is equivalent to a convolution layer with a 1×1 receptive field, and several work have explored convnet architectures with small receptive fields. For instance, the VGG model [53] uses 3×3 convolutions, and later, other architectures such as the ResNext [54] or

the Xception [55] mix 1×1 and 3×3 convolutions. In contrast to convnets, in our model interaction between patches is obtained via a linear layer that is shared across channels, and that relies on absolute rather than relative positions.

More recently, transformers have emerged as a promising architecture for computer vision [4], [56], [57], [58], [59]. In particular, our architecture takes inspiration from the structure used in the Vision Transformer (ViT) [57], and in consequence, shares many components. Our model takes a set of non-overlapping patches as input and passes them through a series of MLP layers that share the same structure as ViT, replacing the self-attention layer with a linear patch interaction layer. Our architecture is subsampling free. This disadvantages it in FLOPs but makes it more efficient in memory consumption as highlighted by Sandler et al. [60]. Both layers have a global field-of-view, unlike convolutional layers. Whereas in self-attention the weights to aggregate information from other patches are data dependent through queries and keys, in ResMLP the weights are not data dependent and only based on absolute positions of patches. In our implementation we follow the improvements of DeiT [4] to train vision transformers, use the skip-connections from ResNets [9] with pre-normalization of the layers [22], [61].

Finally, our work questions the importance of self-attention in existing architectures. Similar observations have been made in natural language processing. Notably, Synthesizer [62] shows that dot-product self-attention can be replaced by a feedforward network, with competitive performance on sentence representation benchmarks. As opposed to our work, Synthesizer does use data dependent weights, but in contrast to transformers the weights are determined from the queries only.

*Concurrent Works.*The MLP-Mixer model by Tolstikhin et al. [63] brings complementary insights to ours: they achieve interesting performance with larger MLP models pre-trained on the larger public ImageNet-22k and even more data with the proprietary JFT-300M. In contrast, we focus on faster models trained on ImageNet-1k, and provide more architectural ablations. Another concurrent related work includes the work of Melas-Kyriazi [64] and RepMLP [65], which are similar. Their architectures correspond to the second row in the Patch communication ablation in our Table 3, up to the normalization layers. In comparison to our approach, MLP-Mixer uses a MLP on the spatial dimension instead of a simple linear layer in our case. They use a Layer-Normalization instead of just an affine transformation for ResMLP. In RepMLP the authors re-parameterize convolutions into fully-connected layers. Hence they train a convolutional network and reparametrized the networks after training, which substantially depart from our work.

5 Conclusion

In this paper we have shown that a simple residual architecture, whose residual blocks consist of a one-hidden layer feed-forward network and a linear patch interaction layer, achieves an unexpectedly high performance on ImageNet classification benchmarks, provided that we adopt a modern training strategy such as those recently introduced for transformer-based architectures. Thanks to their simple structure, with linear layers as the main mean of communication between patches, we can vizualize the filters learned by this simple MLP. While some of the layers are similar to convolutional filters, we also observe sparse long-range interactions as early as the second layer of the network. We hope that our model free of spatial priors will contribute to further understanding of what networks with less priors learn, and potentially guide the design choices of future networks without the pyramidal design prior adopted by most convolutional neural networks.

Acknowledgments

We would like to thank Mark Tygert for relevant references. This work builds upon the timm library [66] by Ross Wightman.

References

[1] A. Vaswani et al., "Attention is all you need," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 6000–6010.
[2] O. Russakovsky et al., "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015.
[3] A. Dosovitskiy et al., "An image is worth 16x16 words: Transformers for image recognition at scale," in *Proc. Int. Conf. Learn. Representations*, 2021.
[4] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou, "Training data-efficient image transformers & distillation through attention," 2020, *arXiv:2012.12877*.
[5] D. Hendrycks and K. Gimpel, "Gaussian error linear units (GELUs)," 2016, *arXiv:1606.08415*.
[6] H. Touvron, M. Cord, A. Sablayrolles, G. Synnaeve, and H. Jégou, "Going deeper with image transformers," 2021, *arXiv:2103.17239*.
[7] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," 2014, *arXiv:1409.0473*.
[8] M. Caron et al., "Emerging properties in self-supervised vision transformers," 2021, *arXiv:2104.14294*.
[9] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.
[10] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," 2016, *arXiv:1607.06450*.
[11] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. 32nd Int. Conf. Mach. Learn.*, 2015, pp. 448–456.
[12] L. Beyer, O. J. Hénaff, A. Kolesnikov, X. Zhai, and A. van den Oord, "Are we done with ImageNet?" 2020, *arXiv:2006.07159*.
[13] B. Recht, R. Roelofs, L. Schmidt, and V. Shankar, "Do ImageNet classifiers generalize to ImageNet?," in *Proc. 36th Int. Conf. Mach. Learn.*, 2019, pp. 5389–5400.
[14] Y. You et al., "Large batch optimization for deep learning: Training BERT in 76 minutes," in *Proc. Int. Conf. Learn. Representations*, 2020.
[15] I. Radosavovic, R. P. Kosaraju, R. B. Girshick, K. He, and P. Dollár, "Designing network design spaces," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 10425–10433.
[16] A. Brock, S. De, S. L. Smith, and K. Simonyan, "High-performance large-scale image recognition without normalization," 2021, *arXiv:2102.06171*.
[17] M. Tan and Q. V. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," 2019, *arXiv:1905.11946*.
[18] F. Radenović, G. Tolias, and O. Chum, "Fine-tuning CNN image retrieval with no human annotation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 7, pp. 1655–1668, Jul. 2019.
[19] S. d'Ascoli, L. Sagun, J. Bruna, and G. Biroli, "Finding the needle in the haystack with convolutions: On the benefits of architectural bias," in *Proc. 33rd Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 9334–9345.
[20] A. Fan et al., "Training with quantization noise for extreme model compression," in *Proc. Int. Conf. Learn. Representations*, 2021.
[21] A. Défossez, Y. Adi, and G. Synnaeve, "Differentiable model compression via pseudo quantization noise," 2021, *arXiv:2104.09987*.
[22] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," 2016, *arXiv:1603.05027*.
[23] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proc. 14th Int. Conf. Mach. Learn.*, 2011, pp. 315–323.
[24] A. Howard et al., "Searching for MobileNetV3," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 1314–1324.
[25] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. 25th Int. Conf. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.
[26] I. Bello et al., "Revisiting ResNets: Improved training and scaling strategies," 2021, *arXiv:2103.07579*.
[27] A. Krizhevsky, "Learning multiple layers of features from tiny images," Master's thesis, University of Toronto, Toronto, Canada, 2009.
[28] M.-E. Nilsback and A. Zisserman, "Automated flower classification over a large number of classes," in *Proc. 6th Indian Conf. Comput. Vis. Graph. Image Process.*, 2008, pp. 722–729.
[29] J. Krause, M. Stark, J. Deng, and L. Fei-Fei, "3D object representations for fine-grained categorization," in *Proc. 4th Int. IEEE Workshop 3D Representation Recognit.*, 2013, pp. 554–561.
[30] G. V. Horn et al., "The iNaturalist species classification and detection dataset," 2017, *arXiv:1707.06642*.
[31] H. Touvron, A. Sablayrolles, M. Douze, M. Cord, and H. Jégou, "Grafit: Learning fine-grained image representations with coarse labels," 2020, *arXiv:2011.12982*.
[32] M. Ott, S. Edunov, D. Grangier, and M. Auli, "Scaling neural machine translation," in *Proc. 3rd Conf. Mach. Transl.*, 2018, pp. 1–9. [Online]. Available: https://www.aclweb.org/anthology/W18–6301
[33] Y. Wu et al., "Google's neural machine translation system: Bridging the gap between human and machine translation," 2016, *arXiv:1609.08144*.

[34] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, "Convolutional sequence to sequence learning," in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, pp. 1243–1252. [Online]. Available: http://proceedings.mlr.press/v70/gehring17a.html
[35] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba, "Semantic understanding of scenes through the ADE20K dataset," *Int. J. Comput. Vis.*, vol. 127, pp. 302–321, 2018.
[36] T. Xiao, Y. Liu, B. Zhou, Y. Jiang, and J. Sun, "Unified perceptual parsing for scene understanding," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 432–448.
[37] H. Bao, L. Dong, and F. Wei, "BEiT: BERT pre-training of image transformers," 2021, *arXiv:2106.08254*.
[38] A. El-Nouby et al., "XCiT: Cross-covariance image transformers," 2021, *arXiv:2106.09681*.
[39] D. C. Cireşan, U. Meier, L. M. Gambardella, and J. Schmidhuber, "Deep big multilayer perceptrons for digit recognition," in *Neural Networks: Tricks of the Trade*, Berlin, Germany: Springer, 2012, pp. 581–598.
[40] P. Y. Simard et al., "Best practices for convolutional neural networks applied to visual document analysis," in *Proc. 7th Int. Conf. Document Anal. Recognit.*, 2003, pp. 958–963.
[41] C. Chatelain, "Extraction de séquences numériques dans des documents manuscrits quelconques," Ph.D. dissertation, Université de Rouen, Mont-Saint-Aignan, France, 2006.
[42] T. Bluche, "Deep neural networks for large vocabulary handwritten text recognition," Ph.D. dissertation, Université Paris-Sud, Bures-sur-Yvette, France, 2015.
[43] Z. Lin, R. Memisevic, and K. Konda, "How far can we go without convolution: Improving fully-connected networks," 2015, *arXiv:1511.02580*.
[44] D. C. Mocanu, E. Mocanu, P. Stone, P. H. Nguyen, M. Gibescu, and A. Liotta, "Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science," *Nature Commun.*, vol. 9, no. 1, pp. 1–12, 2018.
[45] G. Urban et al., "Do deep convolutional nets really need to be deep and convolutional?," in *Proc. Int. Conf. Learn. Representations*, 2017.
[46] B. Neyshabur, "Towards learning convolutions from scratch," in *Proc. 34th Int. Conf. Neural Inf. Process. Syst.*, 2020, pp. 8078–8088.
[47] S. S. Du, Y. Wang, X. Zhai, S. Balakrishnan, R. Salakhutdinov, and A. Singh, "How many samples are needed to estimate a convolutional neural network?," in *Proc. 32nd Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 371–381.
[48] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, "On large-batch training for deep learning: Generalization gap and sharp minima," in *Proc. Int. Conf. Learn. Representations*, 2017.
[49] R. Novak et al., "Bayesian deep convolutional networks with many channels are Gaussian processes," in *Proc. Int. Conf. Learn. Representations*, 2019.
[50] Y. Khoo and L. Ying, "SwitchNet: A neural network model for forward and inverse scattering problems," *SIAM J. Sci. Comput.*, vol. 41, no. 5, pp. A3182–A3201, 2019.
[51] A. Novikov, D. Podoprikhin, A. Osokin, and D. P. Vetrov, "Tensorizing neural networks," in *Proc. 28th Int. Conf. Neural Inf. Process. Syst.*, 2015, pp. 442–450.
[52] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[53] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Learn. Representations*, 2015.
[54] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 5987–5995.
[55] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 1800–1807.
[56] R. Child, S. Gray, A. Radford, and I. Sutskever, "Generating long sequences with sparse transformers," 2019, *arXiv:1904.10509*.
[57] A. Dosovitskiy, J. T. Springenberg, M. A. Riedmiller, and T. Brox, "Discriminative unsupervised feature learning with convolutional neural networks," in *Proc. 27th Int. Conf. Neural Inf. Process. Syst.*, 2014, pp. 766–774.
[58] N. Parmar et al., "Image transformer," in *Proc. 35th Int. Conf. Mach. Learn.*, 2018, pp. 4055–4064.
[59] H. Zhao, J. Jia, and V. Koltun, "Exploring self-attention for image recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 10073–10082.
[60] M. Sandler, J. Baccash, A. Zhmoginov, and A. G. Howard, "Non-discriminative data or weak model? On the relative importance of data and model resolution," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshop*, 2019, pp. 1036–1044.
[61] M. X. Chen et al., "The best of both worlds: Combining recent advances in neural machine translation," in *Proc. 56th Annu. Meeting Assoc. Comput. Linguistics*, 2018, pp. 76–86.
[62] Y. Tay, D. Bahri, D. Metzler, D.-C. Juan, Z. Zhao, and C. Zheng, "Synthesizer: Rethinking self-attention in transformer models," 2020, *arXiv:2005.00743*.
[63] I. Tolstikhin et al., "MLP-Mixer: An all-MLP architecture for vision," 2021, *arXiv:2105.01601*.
[64] L. Melas-Kyriazi, "Do you even need attention? A stack of feed-forward layers does surprisingly well on ImageNet," 2021, *arXiv:2105.02723*.
[65] X. Ding, X. Zhang, J. Han, and G. Ding, "RepMLP: Re-parameterizing convolutions into fully-connected layers for image recognition," 2021, *arXiv:2105.01883*.
[66] R. Wightman, "Pytorch image models," 2019. [Online]. Available: https://github.com/rwightman/pytorch-image-models
[67] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo, "CutMix: Regularization strategy to train strong classifiers with localizable features," 2019, *arXiv:1905.04899*.
[68] H. Zhang, M. Cissé, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," 2017, *arXiv:1710.09412*.
[69] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang, "Random erasing data augmentation," in *Proc. 34th AAAI Conf. Artif. Intell.*, 2020, pp. 13001–13008.
[70] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le, "RandAugment: Practical automated data augmentation with a reduced search space," 2019, *arXiv:1909.13719*.
[71] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger, "Deep networks with stochastic depth," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 646–661.
[72] E. Hoffer, T. Ben-Nun, I. Hubara, N. Giladi, T. Hoefler, and D. Soudry, "Augment your batch: Improving generalization through instance repetition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 8126–8135.