ResMLP: Feedforward networks for image classification with data-efficient training

Hugo Touvron[1,2] Piotr Bojanowski[1] Mathilde Caron[1,3]
Matthieu Cord[2,4] Alaaeldin El-Nouby[1,3] Edouard Grave[1] Gautier Izacard[1]
Armand Joulin[1] Gabriel Synnaeve[1] Jakob Verbeek[1] Hervé Jégou[1]

[1]Facebook AI [2]Sorbonne University [3]Inria [4]valeo.ai

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.

1 Introduction

Recently, the transformer architecture [60], 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 [50] when pre-trained with a sufficiently large amount of data [16]. 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 [16, 56], 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 Figure 1 and detail it further in Section 2.

like token mixer or channel mixer

The ResMLP architecture is strongly inspired by the vision transformers (ViT) [16], 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 [25]. We observe that the training of ResMLP is more stable than ViTs when using the same training scheme as in DeiT [56] and CaiT [57], allowing to remove the need for batch-specific or cross-channel normalizations such as BatchNorm, GroupNorm or LayerNorm. 推测 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

Preprint. Under review.

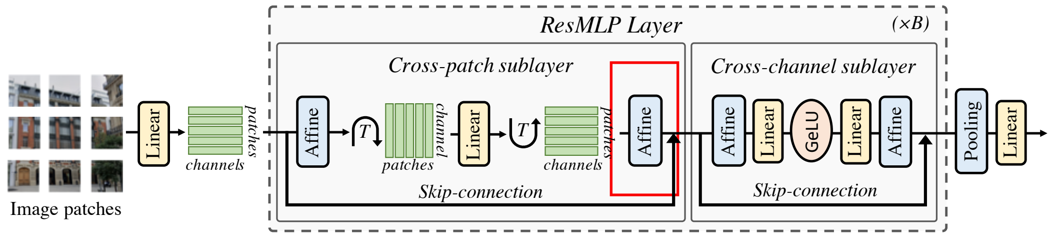

Figure 1: **The ResMLP architecture.** After linearly projecting the image patches into high dimensional embeddings, ResMLP sequentially processes them with (1) a cross-patch linear sublayer; (2) a cross-channel two-layer MLP. The MLP is the same as the FCN sublayer of a Transformer. Each sublayer has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

流露|揭示
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 [2]. This model is similar to the original seq2seq Transformer with ResMLP layers instead of Transformer layers [60]. 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, we make the following observations:

- despite its simplicity, ResMLP reaches surprisingly good accuracy/complexity trade-offs with ImageNet-1k training only[1], without requiring normalization based on batch or channel statistics;
- these models benefit significantly from distillation methods [56]; they are also compatible with modern self-supervised learning methods based on data augmentation, such as DINO [7];
- 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 Figure 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 the reader to Dosovitskiy *et al.* [16] 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 paralleled with a skip-connection [23]. The absence of self-attention layers makes the training more stable, allowing us to replace the Layer Normalization [1] by a simpler Affine transformation:

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

[1]Concurrent work by Tolstikhin *et al.* [55] 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. Other concurrent related work includes that of Melas-Kyriazi [39] and the RepMLP [15] and gMLP [38] models.

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 投入到|吸收到|全神贯注于 absorbed in the adjacent 相邻的 linear layer. Second, as opposed to BatchNorm [30] 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.* [57], 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 [57].

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:

$$\mathbf{Z} = \mathbf{X} + \texttt{Aff}\left(\left(\mathbf{A}\,\texttt{Aff}\left(\mathbf{X}\right)^\top\right)^\top\right), \tag{2}$$

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

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 [25]. 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 [16]. 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 [57]. 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.* [60], 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:

Table 1: **Comparison between architectures on ImageNet classification.** 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.

	Arch.	#params ($\times10^6$)	throughput (im/s)	FLOPS ($\times10^9$)	Peak Mem (MB)	Top-1 Acc.
State of the art	CaiT-M48↑448Υ [57]	356	5.4	329.6	5477.8	86.5
	NfNet-F6 SAM [6]	438	16.0	377.3	5519.3	86.5
Convolutional networks	EfficientNet-B3 [53]	12	661.8	1.8	1174.0	81.1
	EfficientNet-B4 [53]	19	349.4	4.2	1898.9	82.6
	EfficientNet-B5 [53]	30	169.1	9.9	2734.9	83.3
	RegNetY-4GF [47]	21	861.0	4.0	568.4	80.0
	RegNetY-8GF [47]	39	534.4	8.0	841.6	81.7
	RegNetY-16GF [47]	84	334.7	16.0	1329.6	82.9
Transformer networks	DeiT-S [56]	22	940.4	4.6	217.2	79.8
	DeiT-B [56]	86	292.3	17.5	573.7	81.8
	CaiT-XS24 [57]	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

- *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.* [7] 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.* [56] to guide the supervised training of ResMLP with a convnet.

3.1 Experimental setting

Datasets. We train our models on the ImageNet-1k dataset [50], that contains 1.2M images evenly spread over 1,000 object categories. In 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 [4] and ImageNet-v2 [49] datasets. We also report a few data-points when training on ImageNet-21k. Our hyper-parameters are mostly adopted from Touvron et al. [56, 57].

Hyper-parameter settings. In the case of supervised learning, we train our network with the Lamb optimizer [64] 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 [57]. The rest of the hyper-parameters follow the default setting used in DeiT [56]. For the knowledge distillation paradigm, we use the same RegNety-16GF [48] 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

Table 2: **Self-supervised learning** with DINO [7]. Classification accuracy on ImageNet-1k val. ResMLPs evaluated with linear and k-NN evaluation on ImageNet are comparable to convnets but inferior to ViT.

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

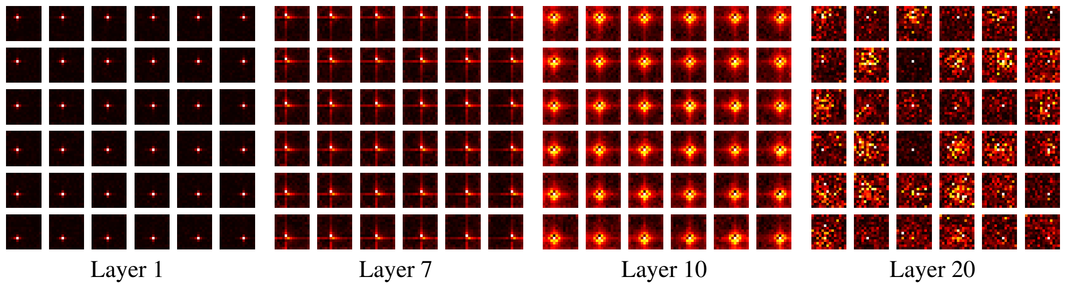

Figure 2: **Visualisation of the linear layers in ResMLP-S24.** For each layer we visualise the rows of the matrix $\mathbf{A}$ as a set of 14×14 pixel images, for sake of space we only show the rows corresponding to the 6×6 central patches. We observe patterns in the linear layers that share similarities with convolutions. In appendix B we provide comparable visualizations for all layers of a ResMLP-S12 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 [7] 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 network 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.* [56]. 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.* [14], 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

Visualisations of the cross-patch sublayers. In Figure 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 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 Figure 3. We measure the sparsity of the matrix $\mathbf{A}$, and compare it to the sparsity of $\mathbf{B}$ and $\mathbf{C}$ from the per-patch

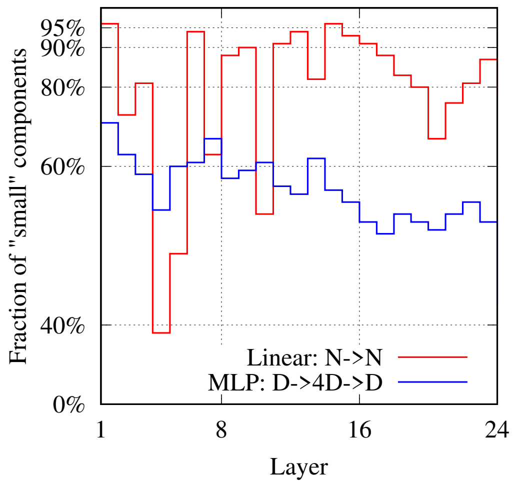

Figure 3: **Sparsity of linear interaction layers.** For each layer (linear and MLP), we show the rate of components whose absolute value is lower than 5% of the maximum. Linear interaction layers are sparser than the matrices involved in the per-patch MLP.

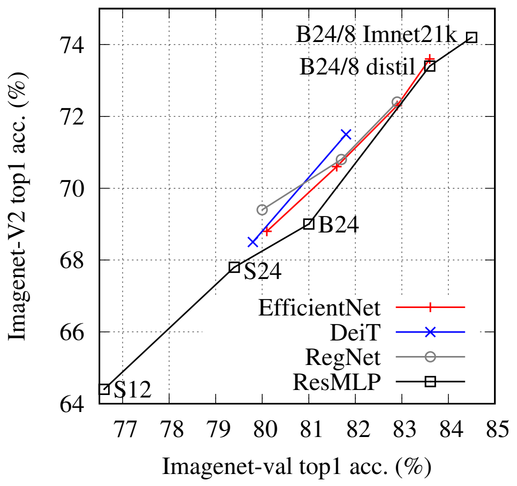

Figure 4: **Top-1 accuracy on ImageNet-V2 *vs.* ImageNet-val.** ResMLPs tend to overfit slightly more under identical training method. This is partially alleviated with by introducing more regularization (more data or distillation, see e.g., ResMLP-B24/8-distil).

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 Figure 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 [19]. The sparsity structure, in particular in earlier layers, see Figure. 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 $\rightarrow$ 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 $\rightarrow$ MLP), which we analyze in more details in the supplementary 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 between ResMLP-S12 and ResMLP-S24 – and requires fewer parameters than a residual MLP block.

The visualization in Figure 2 indicates that many linear interaction layers look like convolutions. 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 at a comparable number of parameters and FLOPs. While full 3×3 convolutions yield best results, they come with 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.

Ablation	Model	Patch size	Params $\times 10^6$	FLOPs $\times 10^9$	Variant	top-1 acc. on ImageNet val	real [4]	v2 [49]
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` $\rightarrow$ Layernorm	77.7	84.1	65.7
Pooling	ResMLP-S12	16	17.7	3.0	average pooling $\rightarrow$ Class-MLP	77.5	84.0	66.1
Patch communication	ResMLP-S12	16	14.9	2.8	linear $\rightarrow$ none	56.5	63.4	43.1
	ResMLP-S12	16	18.6	4.3	linear $\rightarrow$ MLP	77.3	84.0	65.7
	ResMLP-S12	16	30.8	6.0	linear $\rightarrow$ conv 3x3	77.3	84.4	65.7
	ResMLP-S12	16	14.9	2.8	linear $\rightarrow$ conv 3x3 depth-wise	76.3	83.4	64.6
	ResMLP-S12	16	16.7	3.2	linear $\rightarrow$ conv 3x3 depth-separable	77.0	84.0	65.5
Patch size	ResMLP-S12/14	14	15.6	4.0	patch size $16\times16\rightarrow14\times14$	76.9	83.7	65.0
	ResMLP-S12/8	8	22.1	14.0	patch size $16\times16\rightarrow8\times8$	79.1	85.2	67.2
	ResMLP-B24/8	8	129.1	100.2	patch size $16\times16\rightarrow8\times8$	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

Table 3: **Ablation.** Our default configurations are presented in the three first rows. By default we train during 400 epochs. The "old-fashioned" is similar to what was employed for ResNet [23]: SGD, 90-epochs waterfall schedule, same augmentations up to variations due to library used.

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.

Control of overfitting. Since MLPs are subject to overfitting, we show in Fig. 4 a 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 [1], 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 [24], we observed that both choices converged. Pre-normalization in conjunction with Batch Normalization could provide an accuracy gain in some cases, see Appendix A.

We choose to use a GELU [25] function. In Appendix A we also analyze the activation function: ReLU [22] 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 [25] and HardSwish [28].

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 employed for ResNet [23][2]. This is in line with recent work pointing out the importance of the training strategy over the model choice [3, 48]. 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%.

[2]Interestingly, if trained with this "old-fashion" setting, ResMLP-S12 outperforms AlexNet [35] by a margin.

Architecture	FLOPs	Res.	CIFAR_{10}	CIFAR_{100}	Flowers102	Cars	iNat_{18}	iNat_{19}
EfficientNet-B7 [53]	37.0B	600	98.9	91.7	98.8	94.7	_	_
ViT-B/16 [16]	55.5B	384	98.1	87.1	89.5	_	_	_
ViT-L/16 [16]	190.7B	384	97.9	86.4	89.7	_	_	_
Deit-B/16 [56]	17.5B	224	99.1	90.8	98.4	92.1	73.2	77.7
ResNet50 [58]	4.1B	224	_	_	96.2	90.0	68.4	73.7
Grafit/ResNet50 [58]	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

Table 4: **Evaluation on transfer learning.** 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.

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. 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.

3.5 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 [34], Flowers-102 [42], Stanford Cars [33] and iNaturalist [27]. 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.

3.6 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.* [45]. 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.* [60], 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.

Table 5: **Machine translation** on WMT 2014 translation tasks. We report tokenized BLEU on *newstest2014*.

Models	GNMT [62]	ConvS2S [21]	Transf. (base) [60]	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

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 [12, 51], keyword spotting [8] and handwritting recognition [5]. Several works [37, 40, 59] have questioned if FCNs are also competitive on natural image datasets, such as CIFAR-10 [34]. More recently, d'Ascoli *et al.* [14] have shown that a FCN initialized with the weights of a pretrained convnet achieves performance that are superior than the original convnet. Neyshabur [41] 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 [18] and energy landscape [31]. 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 [43], or for inverse scattering problems [32]. More interestingly, the Tensorizing Network [44] 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 pre-defined 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 [35, 36] or transformers [60]. 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 [52] uses 3×3 convolutions, and later, other architectures such as the ResNext [63] or the Xception [11] 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 [10, 17, 46, 56, 67]. In particular, our architecture takes inspiration from the structure used in the Vision Transformer (ViT) [17], and as 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. 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 [56] to train vision transformers, use the skip-connections from ResNets [23] with pre-normalization of the layers [9, 24].

Finally, our work questions the importance of self-attention in existing architectures. Similar observations have been made in natural language processing. Notably, Synthesizer [54] 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.

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.

6 Acknowledgments

We would like to thank Mark Tygert for relevant references. This work builds upon the Timm library [61] by Ross Wightman.

References

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 2, 7

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014. 2

[3] Irwan Bello, W. Fedus, Xianzhi Du, E. D. Cubuk, A. Srinivas, Tsung-Yi Lin, Jonathon Shlens, and Barret Zoph. Revisiting ResNets: Improved training and scaling strategies. *arXiv preprint arXiv:2103.07579*, 2021. 7

[4] Lucas Beyer, Olivier J. Hénaff, Alexander Kolesnikov, Xiaohua Zhai, and Aaron van den Oord. Are we done with ImageNet? *arXiv preprint arXiv:2006.07159*, 2020. 4, 7

[5] Théodore Bluche. *Deep neural networks for large vocabulary handwritten text recognition*. PhD thesis, Université Paris-Sud, 2015. 9

[6] A. Brock, Soham De, S. L. Smith, and K. Simonyan. High-performance large-scale image recognition without normalization. *arXiv preprint arXiv:2102.06171*, 2021. 4

[7] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. *arXiv preprint arXiv:2104.14294*, 2021. 2, 4, 5

[8] Clément Chatelain. *Extraction de séquences numériques dans des documents manuscrits quelconques*. PhD thesis, Université de Rouen, 2006. 9

[9] Mia Xu Chen, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, George Foster, Llion Jones, Niki Parmar, Mike Schuster, Zhifeng Chen, et al. The best of both worlds: Combining recent advances in neural machine translation. In *Annual Meeting of the Association for Computational Linguistics*, 2018. 9

[10] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019. 9

[11] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Conference on Computer Vision and Pattern Recognition*, 2017. 9

[12] Dan Claudiu Cireşan, Ueli Meier, Luca Maria Gambardella, and Jürgen Schmidhuber. Deep big multilayer perceptrons for digit recognition. In *Neural networks: tricks of the trade*, pages 581–598. Springer, 2012. 9

[13] Ekin D. Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V. Le. RandAugment: Practical automated data augmentation with a reduced search space. *arXiv preprint arXiv:1909.13719*, 2019. VI

[14] Stéphane d'Ascoli, Levent Sagun, Joan Bruna, and Giulio Biroli. Finding the needle in the haystack with convolutions: on the benefits of architectural bias. In *NeurIPS*, 2019. 5, 9

[15] Xiaohan Ding, Xiangyu Zhang, Jungong Han, and Guiguang Ding. RepMLP: Re-parameterizing convolutions into fully-connected layers for image recognition. *arXiv preprint arXiv:2105.01883*, 2021. 2

[16] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. 1, 2, 3, 8

[17] Alexey Dosovitskiy, Jost Tobias Springenberg, Martin A. Riedmiller, and Thomas Brox. Discriminative unsupervised feature learning with convolutional neural networks. In *NeurIPS*, 2014. 9

[18] Simon S Du, Yining Wang, Xiyu Zhai, Sivaraman Balakrishnan, Ruslan Salakhutdinov, and Aarti Singh. How many samples are needed to estimate a convolutional neural network? In *NeurIPS*, 2018. 9

[19] Alexandre Défossez, Yossi Adi, and Gabriel Synnaeve. Differentiable model compression via pseudo quantization noise. *arXiv preprint arXiv:2104.09987*, 2021. 6

[20] Angela Fan, Pierre Stock, Benjamin Graham, Edouard Grave, Rémi Gribonval, Hervé Jégou, and Armand Joulin. Training with quantization noise for extreme model compression. In *International Conference on Learning Representations*, 2021. 6

[21] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. Convolutional sequence to sequence learning. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1243–1252. PMLR, 06–11 Aug 2017. 8

[22] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *International Conference on Machine Learning*, 2011. 7

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Conference on Computer Vision and Pattern Recognition*, 2016. 2, 7, 9, V

[24] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. *arXiv preprint arXiv:1603.05027*, 2016. 7, 9

[25] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (GELUs). *arXiv preprint arXiv:1606.08415*, 2016. 1, 3, 7

[26] Elad Hoffer, Tal Ben-Nun, Itay Hubara, Niv Giladi, Torsten Hoefler, and Daniel Soudry. Augment your batch: Improving generalization through instance repetition. In *Conference on Computer Vision and Pattern Recognition*, 2020. VI

[27] Grant Van Horn, Oisin Mac Aodha, Yang Song, Alexander Shepard, Hartwig Adam, Pietro Perona, and Serge J. Belongie. The iNaturalist species classification and detection dataset. *arXiv preprint arXiv:1707.06642*, 2017. 8

[28] A. Howard, Mark Sandler, G. Chu, Liang-Chieh Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Quoc V. Le, and H. Adam. Searching for MobileNetV3. *International Conference on Computer Vision*, 2019. 7

[29] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q. Weinberger. Deep networks with stochastic depth. In *European Conference on Computer Vision*, 2016. VI

[30] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, 2015. 3

[31] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. In *International Conference on Learning Representations*, 2017. 9

[32] Yuehaw Khoo and Lexing Ying. Switchnet: a neural network model for forward and inverse scattering problems. *SIAM Journal on Scientific Computing*, 41(5):A3182–A3201, 2019. 9

[33] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, 2013. 8

[34] A. Krizhevsky. Learning multiple layers of features from tiny images. Master's thesis, University of Toronto, 2009. 8, 9

[35] A. Krizhevsky, I. Sutskever, and G. Hinton. ImageNet classification with deep convolutional neural networks. In *NeurIPS*, 2012. 7, 9

[36] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 9

[37] Zhouhan Lin, Roland Memisevic, and Kishore Konda. How far can we go without convolution: Improving fully-connected networks. *arXiv preprint arXiv:1511.02580*, 2015. 9

[38] Hanxiao Liu, Zihang Dai, David R. So, and Quoc V. Le. Pay attention to MLPs. *arXiv preprint arXiv:2105.08050*, 2021. 2

[39] Luke Melas-Kyriazi. Do you even need attention? A stack of feed-forward layers does surprisingly well on ImageNet. *arXiv preprint arXiv:2105.02723*, 2021. 2

[40] Decebal Constantin Mocanu, Elena Mocanu, Peter Stone, Phuong H Nguyen, Madeleine Gibescu, and Antonio Liotta. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. *Nature communications*, 9(1):1–12, 2018. 9

[41] Behnam Neyshabur. Towards learning convolutions from scratch. In *NeurIPS*, 2020. 9

[42] M-E. Nilsback and A. Zisserman. Automated flower classification over a large number of classes. In *Proceedings of the Indian Conference on Computer Vision, Graphics and Image Processing*, 2008. 8

[43] Roman Novak, Lechao Xiao, Jaehoon Lee, Yasaman Bahri, Greg Yang, Jiri Hron, Daniel A Abolafia, Jeffrey Pennington, and Jascha Sohl-Dickstein. Bayesian deep convolutional networks with many channels are Gaussian processes. In *International Conference on Learning Representations*, 2019. 9

[44] Alexander Novikov, Dmitrii Podoprikhin, Anton Osokin, and Dmitry P Vetrov. Tensorizing neural networks. In *Neurips*, 2015. 9

[45] Myle Ott, Sergey Edunov, David Grangier, and Michael Auli. Scaling neural machine translation. In *ACL*, pages 1–9. Association for Computational Linguistics, 2018. 8

[46] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In *International Conference on Machine Learning*, 2018. 9

[47] Filip Radenović, Giorgos Tolias, and Ondrej Chum. Fine-tuning CNN image retrieval with no human annotation. IEEE *Transactions on Pattern Analysis and Machine Intelligence*, 41(7):1655 – 1668, 2018. 4

[48] Ilija Radosavovic, Raj Prateek Kosaraju, Ross B. Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. *Conference on Computer Vision and Pattern Recognition*, 2020. 4, 7

[49] B. Recht, Rebecca Roelofs, L. Schmidt, and V. Shankar. Do ImageNet classifiers generalize to ImageNet? In *International Conference on Machine Learning*, 2019. 4, 7

[50] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. 1, 4

[51] Patrice Y Simard, David Steinkraus, John C Platt, et al. Best practices for convolutional neural networks applied to visual document analysis. In *ICDAR*, 2003. 9

[52] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015. 9

[53] Mingxing Tan and Quoc V. Le. EfficientNet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019. 4, 8

[54] Yi Tay, Dara Bahri, Donald Metzler, Da-Cheng Juan, Zhe Zhao, and Che Zheng. Synthesizer: Rethinking self-attention in transformer models. *arXiv preprint arXiv:2005.00743*, 2020. 9

[55] Ilya Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Daniel Keysers, Jakob Uszkoreit, Mario Lucic, and Alexey Dosovitskiy. MLP-Mixer: An all-MLP architecture for vision. *arXiv preprint arXiv:2105.01601*, 2021. 2

[56] Hugo Touvron, M. Cord, M. Douze, F. Massa, Alexandre Sablayrolles, and H. Jégou. Training data-efficient image transformers & distillation through attention. *arXiv preprint arXiv:2012.12877*, 2020. 1, 2, 4, 5, 8, 9, I, V

[57] Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. Going deeper with image transformers. *arXiv preprint arXiv:2103.17239*, 2021. 1, 3, 4, I

[58] Hugo Touvron, Alexandre Sablayrolles, M. Douze, M. Cord, and H. Jégou. Grafit: Learning fine-grained image representations with coarse labels. *arXiv preprint arXiv:2011.12982*, 2020. 8

[59] Gregor Urban, Krzysztof J Geras, Samira Ebrahimi Kahou, Ozlem Aslan, Shengjie Wang, Rich Caruana, Abdelrahman Mohamed, Matthai Philipose, and Matt Richardson. Do deep convolutional nets really need to be deep and convolutional? In *International Conference on Learning Representations*, 2017. 9

[60] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. 1, 2, 3, 8, 9

[61] Ross Wightman. Pytorch image models. `https://github.com/rwightman/pytorch-image-models`, 2019. 10

[62] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016. 8

[63] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Conference on Computer Vision and Pattern Recognition*, 2017. 9

[64] Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. Large batch optimization for deep learning: Training BERT in 76 minutes. In *International Conference on Learning Representations*, 2020. 4

[65] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. CutMix: Regularization strategy to train strong classifiers with localizable features. *arXiv preprint arXiv:1905.04899*, 2019. VI

[66] Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017. VI

[67] Hengshuang Zhao, Jiaya Jia, and Vladlen Koltun. Exploring self-attention for image recognition. In *Conference on Computer Vision and Pattern Recognition*, 2020. 9

[68] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *AAAI*, 2020. VI

ResMLP: Feedforward networks for image classification with data-efficient training

Appendix

A Report on our exploration phase

As discussed in the main paper, our work on designing a residual multi-layer perceptron was inspired by the Vision Transformer. For our exploration, we have adopted the recent CaiT variant [57] as a starting point. This transformer-based architecture achieves state-of performance with Imagenet-training only (achieving 86.5% top-1 accuracy on Imagenet-val for the best model). Most importantly, the training is relatively stable with increasing depth.

In our exploration phase, our objective was to radically simplify this model. For this purpose, we have considered the Cait-S24 model for faster iterations. This network consists of 24-layer with a working dimension of 384. All our experiments below were carried out with images in resolution 224×224 and $N = 16 \times 16$ patches. Trained with regular supervision, Cait-S24 attains 82.7% top-1 acc. on Imagenet.

SA $\rightarrow$ MLP. The self-attention can be seen a weight generator for a linear transformation on the values. Therefore, our first design modification was to get rid of the self-attention by replacing it by a residual feed-forward network, which takes as input the *transposed* set of patches instead of the patches. In other terms, in this case we alternate residual blocks operating along the channel dimension with some operating along the patch dimension. In that case, the MLP replacing the self-attention consists of the sequence of operations

$$(\cdot)^T \text{ — linear } N \times 4N \text{ — GELU — linear } 4N \times N \text{ — } (\cdot)^T$$

Hence this network is symmetrical in N and d. By keeping the other elements identical to CaiT, the accuracy drops to 80.2% (-2.5%) when replacing self-attention layers.

Class-attention $\rightarrow$ class-MLP. If we further replace the class-attention layer of CaiT by a MLP as described in our paper, then we obtain an attention-free network whose top-1 accuracy on Imagenet-val is 79.2%, which is comparable to a ResNet-50 trained with a modern training strategy. This network has served as our baseline for subsequent ablations. Note that, at this stage, we still include LayerScale, a class embedding (in the class-MLP stage) and positional encodings.

Distillation. The same model trained with distillation inspired by Touvron et al. [56] achieves 81.5%. The distillation variant we choose corresponds to the "hard-distillation", whose main advantage is that it does not require any parameter-tuning compared to vanilla cross-entropy. Note that, in all our experiments, this distillation method seems to bring a gain that is complementary and seemingly almost orthogonal to other modifications.

Activation: LayerNorm $\rightarrow$ X. We have tried different activations on top of the aforementioned MLP-based baseline, and kept GeLU for its accuracy and to be consistent with the transformer choice.

Activation	top-1 acc.
GeLU (baseline)	79.2%
SILU	78.7%
Hard Swish	78.8%
ReLU	79.1%

Ablation on the size of the communication MLP. For the MLP that replaced the class-attention, we have explored different sizes of the latent layer, by adjusting the expansion factor e in the sequence: linear $N \times e \times N$ — GELU — linear $e \times N \times N$. For this experiment we used average pooling to aggregating the patches before the classification layer.

expansion factor $\times e$	$\times$0.25	$\times$0.5	$\times$1	$\times$2	$\times$3	$\times$4
Imnet-val top-1 acc.	78.6	79.2	79.2	79.3	78.8	78.8

We observe that a large expansion factor is detrimental in the patch communication, possibly because we should not introduce too much capacity in this residual block. This has motivated the choice of adopting a simple linear layer of size $N \times N$: This subsequently improved performance to 79.5% in a setting comparable to the table above. Additionally, as shown earlier this choice allows visualizations of the interaction between patches.

Normalization. On top of our MLP baseline, we have tested different variations for normalization layers. We report the variation in performance below.

Pre-normalization	top-1 acc.
Layernorm (baseline)	79.2%
Batch-Norm	+0.8%
ℓ_2-norm	+0.4%
no norm (`Aff`)	+0.4%

For the sake of simplicity, we therefore adopted only the `Aff` transformation so as to not depend on any batch or channel statistics.

Position encoding. In our experiments, removing the position encoding does not change the results when using a MLP or a simple linear layer as a communication mean across patch embeddings. This is not surprising considering that the linear layer implicitly encodes each patch identity as one of the dimension, and that additionally the linear includes a bias that makes it possible to differentiate the patch positions before the shared linear layer.

B Analysis of interaction layers in 12-layer networks

In this section we further analyze the linear interaction layers in 12-layer models.

In Figure B.1 we consider a ResMLP-S12 model trained on the ImageNet-1k dataset, as explained in Section 3.1, and show all the 12 linear patch interaction layers. The linear interaction layers in the supervised 12-layer model are similar to those observed in the 24-layer model in Figure 2.

We also provide the corresponding sparsity measurements for this model in Figure B.2, analogous to the measurements in Figure 3 for the supervised 24-layer model. The sparsity levels in the supervised 12-layer model (left panel) are similar to those observes in the supervised 24-layer model, cf. Figure 3. In the right panel of Figure B.2 we consider the sparsity levels of the Distilled 12-layer model, which are overall similar to those observed for supervised the 12-layer and 24-layer models.

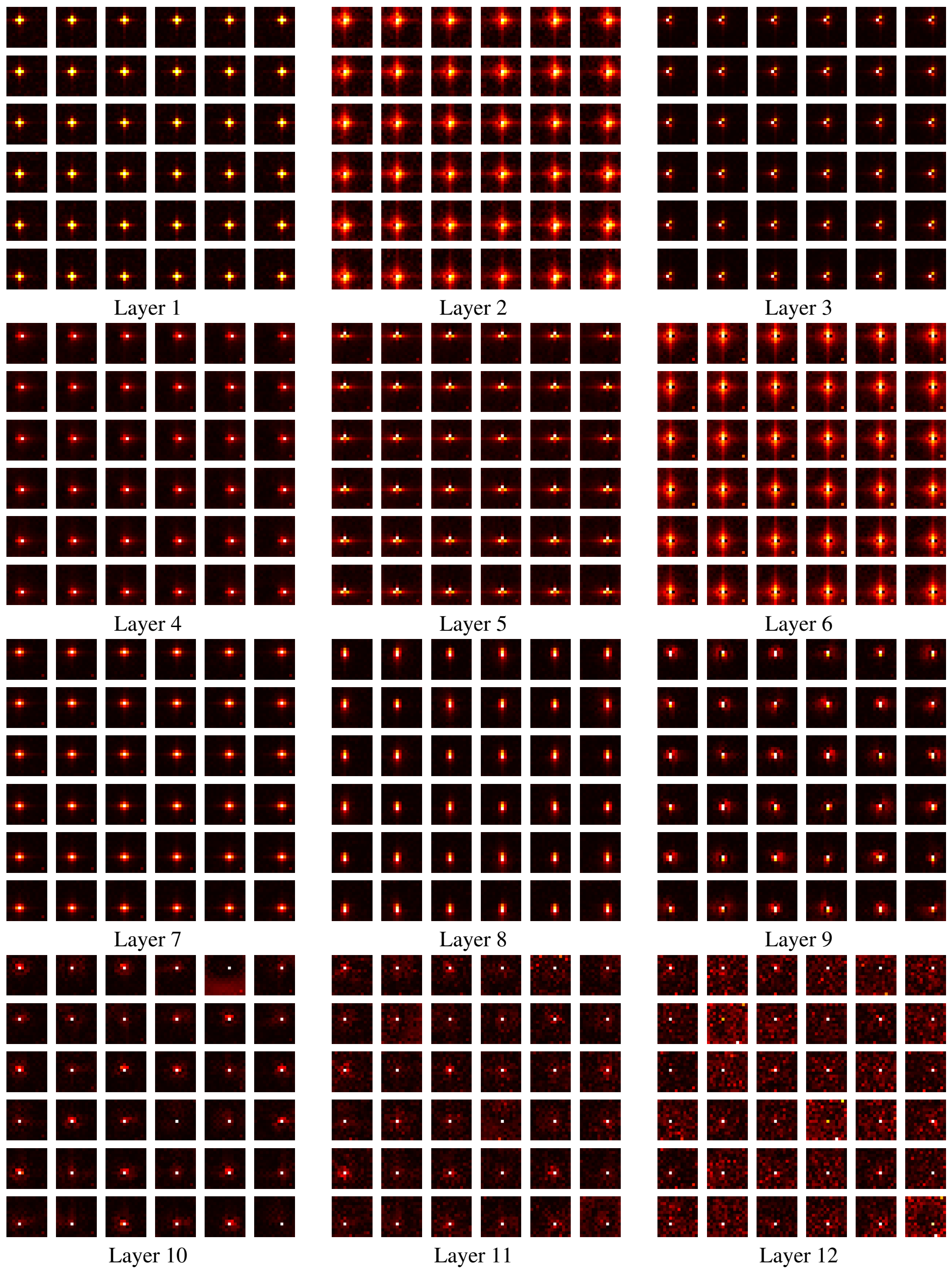

Figure B.1: **Visualisation of the linear interaction layers in the supervised ResMLP-S12 model.** For each layer we visualise the rows of the matrix $\mathbf{A}$ as a set of 14×14 pixel images, for sake of space we only show the rows corresponding to the $6{\times}6$ central patches.

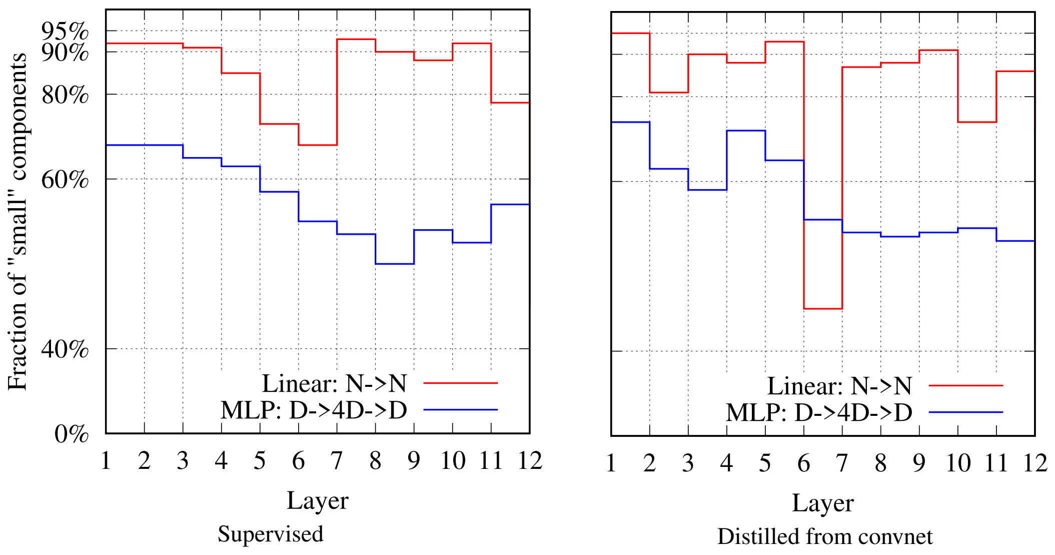

Figure B.2: Degree of sparsity (fraction of values small than 5% of the maximum) for Linear and MLP layers, for ResMLP-S12 networks. The network trained in supervised mode and the one learned with distillation overall have a comparable degree of sparsity. The self-supervised model, trained during 300 epochs *vs.* 400 for the other ones, is less sparse on the patch communication linear layer.

C Model definition in Pytorch

In Algorithm 1 we provide the pseudo-pytorch-code associated with our model.

Algorithm 1 Pseudocode of ResMLP in PyTorch-like style

```
# No norm layer
class Affine(nn.Module):
   def __init__(self, dim):
      super().__init__()
      self.alpha = nn.Parameter(torch.ones(dim))
      self.beta = nn.Parameter(torch.zeros(dim))
   def forward(self, x):
      return self.alpha * x + self.beta

# MLP on channels
class Mlp(nn.Module):
   def __init__(self, dim):
      super().__init__()
      self.fc1 = nn.Linear(dim, 4 * dim)
      self.act = nn.GELU()
      self.fc2 = nn.Linear(4 * dim, dim)
   def forward(self, x):
      x = self.fc1(x)
      x = self.act(x)
      x = self.fc2(x)
      return x

# ResMLP blocks: a linear between patches + a MLP to process them independently
class ResMLP_BLocks(nn.Module):
   def __init__(self, nb_patches ,dim, layerscale_init):
      super().__init__()
      self.affine_1 = Affine(dim)
      self.affine_2 = Affine(dim)
      self.linear_patches = nn.Linear(nb_patches, nb_patches) #Linear layer on patches
      self.mlp_channels = Mlp(dim) #MLP on channels
      self.layerscale_1 = nn.Parameter(layerscale_init * torch.ones((dim))) #LayerScale
      self.layerscale_2 = nn.Parameter(layerscale_init * torch.ones((dim))) # parameters

   def forward(self, x):
      res_1 = self.linear_patches(self.affine_1(x).transpose(1,2)).transpose(1,2)
      x = x + self.layerscale_1 * res_1
      res_2 = self.mlp_channels(self.affine_2(x))
      x = x + self.layerscale_2 * res_2
      return x

# ResMLP model: Stacking the full network
class ResMLP_models(nn.Module):
   def __init__(self, dim, depth, nb_patches, layerscale_init, num_classes):
      super().__init__()
      self.patch_projector = Patch_projector()
      self.blocks = nn.ModuleList([
         ResMLP_BLocks(nb_patches ,dim, layerscale_init)
         for i in range(depth)])
      self.affine = Affine(dim)
      self.linear_classifier = nn.Linear(dim, num_classes)

   def forward(self, x):
      B, C, H, W = x.shape
      x = self.patch_projector(x)
      for blk in self.blocks:
         x = blk(x)
      x = self.affine(x)
      x = x.mean(dim=1).reshape(B,-1) #average pooling
      return self.linear_classifier(x)
```

D Additional Ablations

Training recipe. DeiT [56] proposes a training strategy which allows for data-efficient vision transformers on ImageNet only. In Table D.1 we ablate each component of the DeiT training to go back to the initial ResNet50 training. As to be expected, the training used in the ResNet-50 paper [23] degrades the performance.

Training schedule. Table D.2 compares the performance of ResMLP-S36 according to the number of training epochs. We observe a saturation of the performance after 800 epochs for ResMLP. This saturation is observed in DeiT from 400 epochs. So ResMLP needs more epochs to be optimal.

Ablation	Imagenet1k-val top1-acc (%)
DeiT-style training	76.6
− CutMix [65] - Mixup [66]	76.0
− Random-erasing [68]	75.9
− RandAugment [13]	73.2
SGD optimizer	72.1
− Stochastic-depth [29]	70.7
− Repeated-augmentation [26]	69.4
120 epochs	67.7
Step decay	63.5
Batch size 256	69.3
ResNet-50 training 90 epochs	69.2

Table D.1: Ablations on the training strategy

Epochs	300	400	500	800	1000
Inet-val	79.3	79.7	80.1	**80.4**	80.3
Inet-real	85.5	85.6	**85.9**	85.8	85.7
Inet-V2	68.0	68.4	68.4	68.9	**69.0**

Table D.2: We compare the performance of ResMLP-S36 according to the number of training epochs.

Pooling layers. Table D.3 compares the performance of two pooling layers: average-pooling and class-MLP, with different depth with and without distillation. We can see that class-MLP performs much better than average pooling by changing only a few FLOPs and number of parameters. Nevertheless, the gap seems to decrease between the two approaches with deeper models.

#layers	Params	Flops	Training	Pooling	top-1 acc. on ImageNet		
	$\times 10^6$	$\times 10^9$		layer	Inet-val	Inet-real	Inet-V2
12	15.4	3.0	regular	average	76.6	83.3	64.4
24	30.0	6.0	regular	average	79.4	85.3	67.9
36	44.7	8.9	regular	average	79.7	85.6	68.4
12	17.7	3.0	regular	class-MLP	77.5	84.0	66.1
24	32.4	6.0	regular	class-MLP	79.8	85.6	68.6
36	47.1	8.9	regular	class-MLP	80.5	86.3	69.5
12	15.4	3.0	distillation	average	77.8	84.6	66.0
24	30.0	6.0	distillation	average	80.8	86.6	69.8
36	44.7	8.9	distillation	average	81.0	86.8	70.2
12	17.7	3.0	distillation	class-MLP	78.6	85.2	67.3
24	32.4	6.0	distillation	class-MLP	81.1	87.0	70.4
36	47.1	8.9	distillation	class-MLP	81.4	87.3	70.7

Table D.3: We compare the performance of two pooling layers: average-pooling and class-MLP, with ResMLP-S architecture. We compare different depth, regular training and distillation.