Develop a Proof of Concept REPORT

Project No. 7

2023/04/16

Classification of Dog breeds using the ConvNext model.

OPENCLASSROOMS

Table of Contents

Develop a Proof of Concept Report

Introduction

The field of machine learning, and more generally of data science, is evolving very quickly. It is therefore important to develop the habit of being kept abreast of advances in the field by carrying out a thematic watch. It is also important to be able to quickly build skills on a new theme, knowing how to carry out research and put a new algorithm into practice independently.

Project Description

The objective of this project is to find a newer algorithm to improve the results obtained in the previous project. After reviewing the state of the art in the CNN domain, the new algorithm ConvNext is chosen to be used with the dog breed dataset to evaluate its performance.

This report will be organized as follows: I will first start with a brief description of datasets, then I will approach the ConvNext algorithm with a general and detailed description, then present its implementation and discuss the results obtained.

Data Description

The Stanford Dogs dataset contains images of 120 breeds of dogs from around the world. This dataset has been built using images and annotation from ImageNet for the task of fine-grained image categorization. Contents of this dataset:

- Number of categories: 120
- Number of images: 20,580

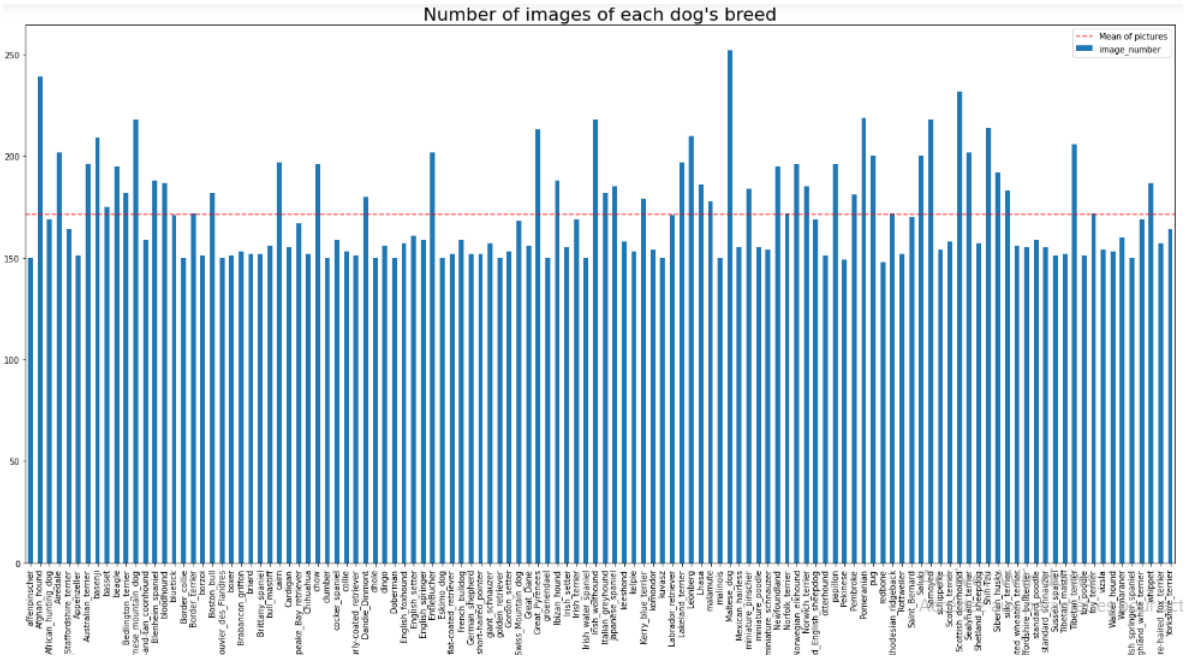

Fig. 1. Number of images of each dog's breed in Stanford Dogs Dataset

For reasons of computer time, and in order to make sense of the comparison with the results of the previous project, I will only use a part of it, namely, the 20 dog breeds most represented in the dataset.

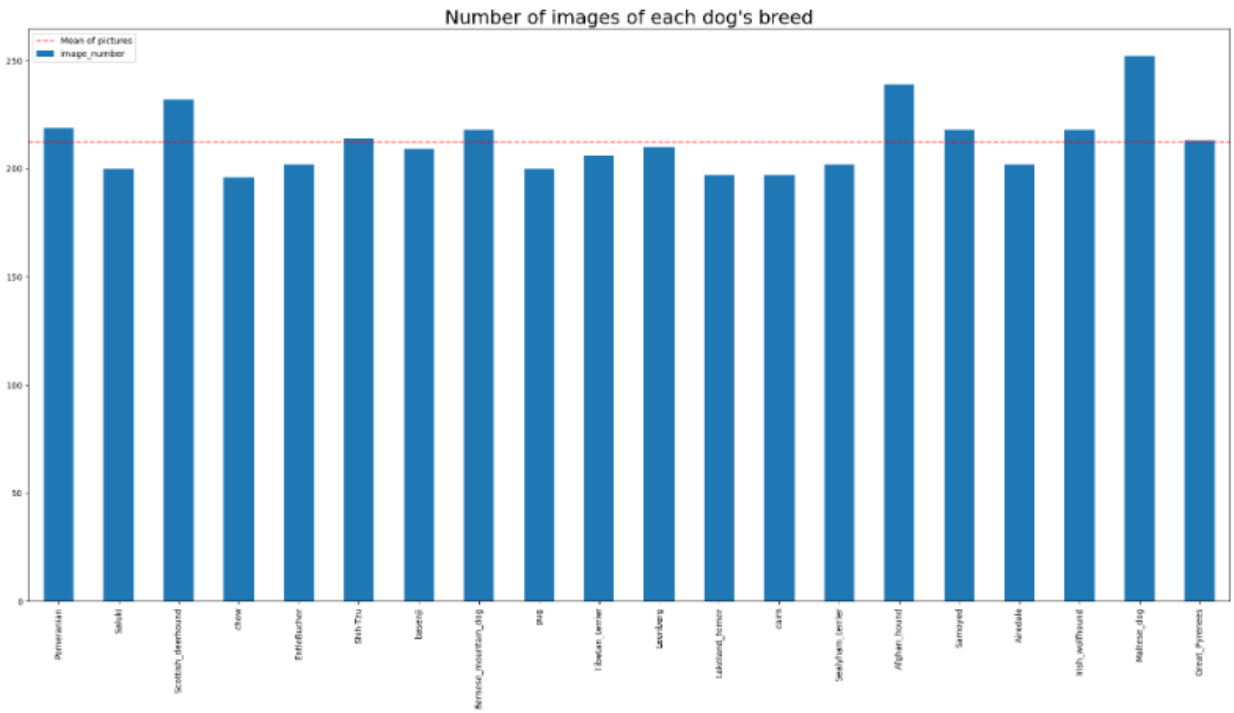

Fig. 2. **Number of images of each dog's breed (20 dog's breed selected)**

NB. No image preprocessing was used for MobileNet, I will also not do any preprocessing.

Model Description

Introduction

For many years, we have used ConvNets as the default model in image classification. But, this changed when Vision transformers, previously introduced to solve problems in Natural Language Processing (NLP), took over the state-of-the-art Convolutional Neural Network in solving image classification tasks. This research is discussed in this paper.

A recent research claims that by borrowing ideas from the successes of the Vision transformer and CNNs, one can build a pure ConvNet whose performance match state-of-the-art-models like the Vision transformer.

This ConvNet takes a standard neural network, a ResNet-50, and morphs it such that the design approaches a vision transformer. Besides, it surpasses the Transformer in terms of performance.

This article will discuss how ConvNets borrows ideas from modern training tricks that Transformers usually benefit from to build the ultimate ConvNext architecture.

Relationship between ConvNets and Transformers

In 2012, the Deep learning renaissance was largely driven by ConvNets. The introduction of AlexNet established the "ImageNet moment" in computer vision.

This moment has rapidly evolved over the years to better and more efficient models such as VGGNet, Inception, ResNeXt, DenseNet, MobileNet, and EfficientNet.

The success of CNNs is based on their ability to share parameters within the image locations, induce translations, and equivariance.

Transformers were introduced for text processing in 2017. This model only had some applications in image generation and image-text understanding.

Due to this, Transformers went unnoticed in the computer vision world. But in the field of Natural Language processing, it was widely used and very successful.

It wasn't until early 2021 when the Google Research team released a paper detailing how Transformers could outperform ConvNets in solving computer vision tasks.

This Transformer model introduced a patch layer that splits an image into a sequence of patches of 16 by 16 pixels. This Transformer model is called the Vision Transformer (ViT).

Yet, the ViT had to rely heavily on a lot of training tricks such as data augmentation to make it reach the performances of state-of-the-art models like ConvNets.

This model faced difficulties when it came to solving more general computer vision tasks. This led to the release of a new vision transformer model.

In mid-2021, a new transformer model called the Swin Transformer was released. It is discussed in this paper.

It introduced sliding windows (used in CNNs), which make them resemble ConvNets pretty much. This model made the vision transformer more general-purpose and could be used for a wide variety of vision tasks.

So, what if we morphed ConvNets towards the Swin Transformer? Will modernizing ConvNets make them stylish enough for the 2020s? Since the style of the 2020s is given by the Transformers, the goal is to make ConvNets more transformer-like.

How do they do this?

Transforming from ConvNet to ConvNeXts

Architecture & Approach

The design process of ConvNeXt was driven by one key question:

How do design decisions in Transformers impact ConvNets' performance?

The authors start with a standard ResNet (e.g. ResNet50) and gradually "*modernize*" the architecture to the construction of a hierarchical vision Transformer (e.g. Swin-T).

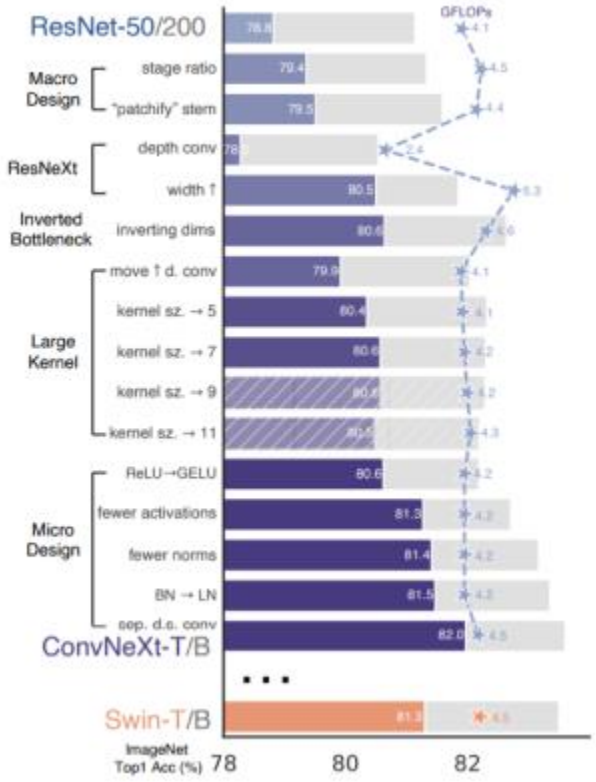

Fig.3. The Design Journey Of ConvNeXt

Training Methodology

Taking a ResNet-50, they train it using similar techniques as the Vision Transformer.

They use the AdamW optimizer, train it using more epochs, apply some heavy data augmentation technique, and regularization. These techniques applied together, increase the performance of ResNet-50 from **76.1%** to **78.8%** on the ImageNet Top1 accuracy.

Redesigning the Macro design of the ResNet

For the stage ratio, they adjusted the number of blocks in each stage which improves the model accuracy. They make the sliding windows in ResNet behave more similarly to the patches of the vision transformer.

So, with a large kernel size and a stride such that the sliding window does not overlap, this looks pretty much like the non-overlapping patches in the transformer. These changes increase the accuracy from **78.8% to 79.4%**.

The idea of inception

ResNeXt has a better FLOPs/accuracy trade-off than a vanilla ResNet. The core component is grouped convolution, where the convolutional filters are separated into different groups. At a high level, ResNeXt's guiding principle is to "*use more groups, expand width*".

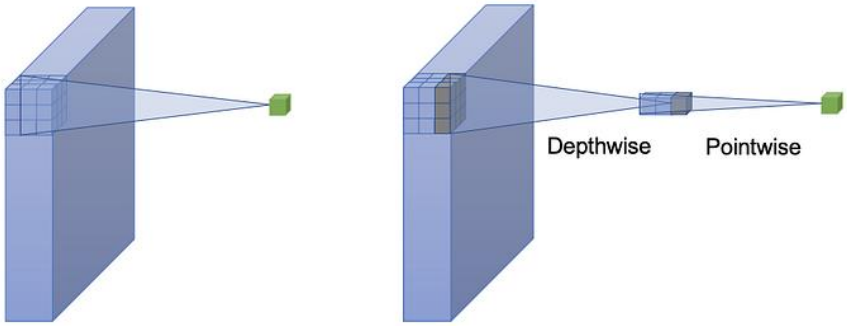

Fig.4. Standard Convolution (left), Depthwise Convolution & Pointwise Convolution (right)

ConvNeXt uses depthwise convolution, a special case of grouped convolution where the number of groups equals the number of channels. Depthwise convolution is similar to the weighted sum operation in self-attention, which operates on a per-channel basis, i.e., only mixing information in the spatial dimension.

Depthwise convolution reduces the network FLOPs, and the accuracy. But following ResNeXt, ConvNeXt increases the network width from 64 to 96, the same number of channels as Swin-T. This brings the network performance to **80.5%** with increased **FLOPs (5.3G**).

The inverted bottleneck

In Swin Transformers, every transformer block creates an inverted bottleneck. The output of four blocks that get concatenated increases the size of the hidden dimensions by four times.

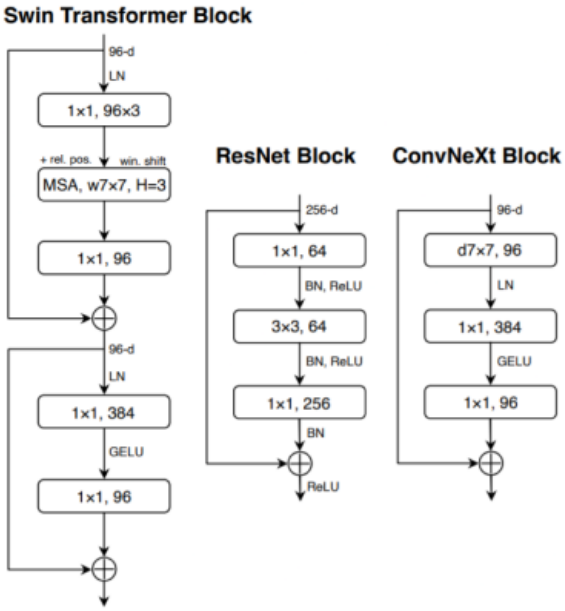

Fig.5 The inverted bottleneck

In ConvNeXts, they copy this idea by devising an inverted bottleneck with an expansion ratio of 4. They find that it increases the performance of the model from **80.5% to 80.6%**.

This idea was initially popularized by MobileNetV2. You can read more about it here.

Increasing the kernel size

Increasing the kernel sizes is essential as it tries to equal the power of vision transformer models with a global receptive field. The vision transformer tends to view the whole image at once through self-attention that spans the entire image.

Swin transformers tend to limit the self-attention window. If the ResNets window size increases, we reach the same compromise.

Micro design choices

ConvNeXt also adopts some mirco scale architecture features from Transformers, i.e., layer level design aspects.

- **Activation function**

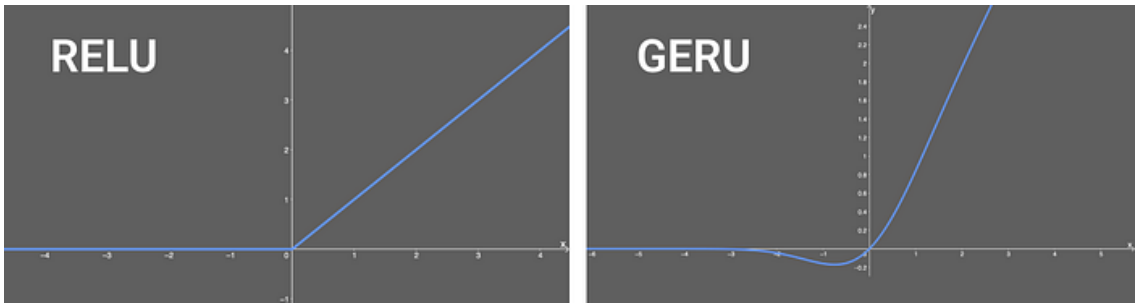

Fig.6 Activation Function

It replaces the ReLU activation function with its smoother variant Gaussian Error Linear Unit, GERU. Replicating the style of a Transfomer block, ConvNeXt drops all the GERU layers from the residual block except the one between two 1x1 Conv layers.

- **Normalization**

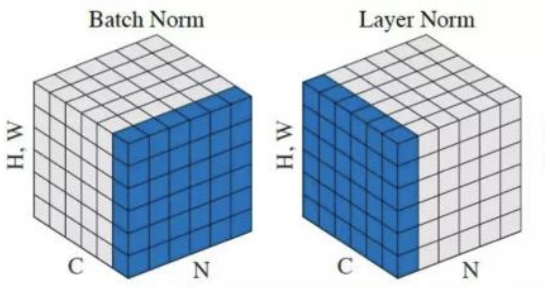

Fig.7 Batch/Layer Normalisation

In addition to activation functions, Transformer blocks have fewer normalization functions as well. ConvNeXt eliminates two normalization layers and leaves only one before the 1x1 Conv layers. And, it replaces the BatchNorm is replaced by the simple Layer Normalization used by Transformers. Lastly,

ConvNeXt adds a separate downsampling layer between stages. It uses 2x2 Conv layers with a stride of 2 for downsampling.

Last Epoch

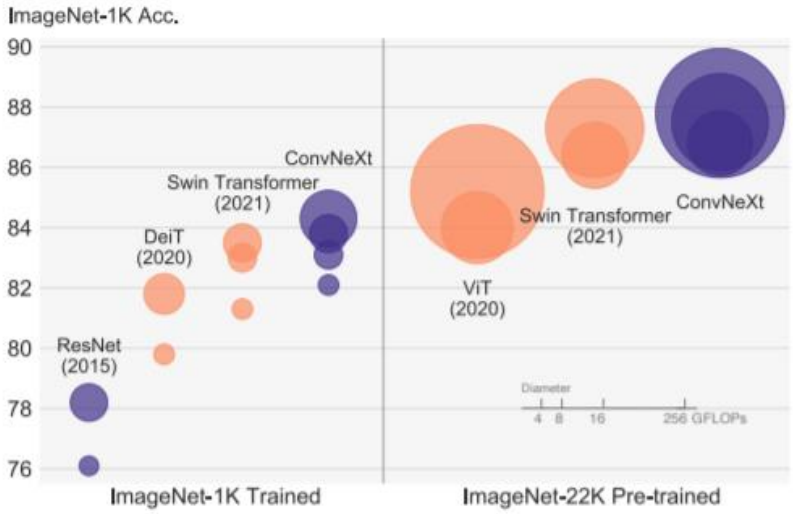

Fig.8 Plot of accuracies of different transformers and convolution architectures

On ImageNet-1K, ConvNeXt competes admiringly with two strong ConvNet baselines — RegNet and EfficientNet in terms of the accuracy-computation trade-off, and, the inference throughputs. It also outperforms Swin Transformer without specialized modules such as shifted windows or relative position bias. Furthermore, ConvNeXts achieves better throughput (**774.7 images/s**) compared to Swin Transformers (**757.9 images/s**).

The general consensus is that transformers have fewer inductive biases and as a result, they perform better than ConvNet on larger scales. But this is refuted by the **87.8%** accuracy of ConvNeXt-Xl on ImageNet-22K beating Swin Transformers.

Implementation & Results

Implementation

The first version was implemented in tensorflow as a module: tf.keras.applications.convnext. This is advantageous because we can use the same code used in the previous project with exactly the same steps, which allows us to have a good comparison in all the steps.

Two versions of ConvNext have been tested:

- ConvNeXtBase
- ConvNeXtTiny

The base, models was first pre-trained on the ImageNet-21k dataset and then fine-tuned on the ImageNet-1k dataset.

Results

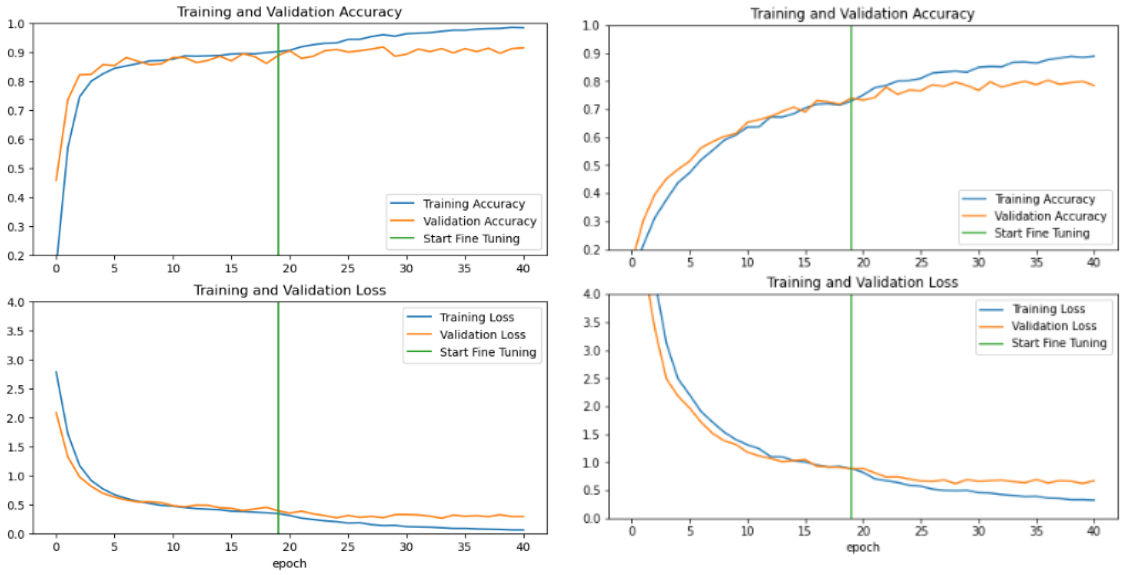

Fig.8 ConvNext Base version (Left) vs MobileNetV2 (right)

Evaluation and prediction

```
Entrée [50]: (loss, accuracy) = model.evaluate(test_ds)
             print('Test accuracy :', accuracy)

5/5 [==============================] - 2s 340ms/step - loss: 0.3241 - accuracy: 0.9187
Test accuracy : 0.918749988079071
```

Fig.9a ConvNext accuracy (Base version)

Evaluation and prediction

```
[45]: (loss, accuracy) = model.evaluate(test_ds)
      print('Test accuracy :', accuracy)

5/5 [==============================] - 2s 234ms/step - loss: 0.5370 - accuracy: 0.8438
Test accuracy : 0.84375
```

Fig.9b ConvNext accuracy (Tiny version)

Evaluation and prediction

```
Entrée [165]: (loss, accuracy) = model.evaluate(test_ds)
              print('Test accuracy :', accuracy)

5/5 [==============================] - 4s 671ms/step - loss: 0.6291 - accuracy: 0.8000
Test accuracy : 0.800000011920929
```

Fig.10 MobileNetV2 accuracy

We see a clear improvement of 10% using ConvNext, going from **80%** to **91%** in terms of accuracy.

Discussion

The ConvNext model has proven to be effective and has outperformed even transformer (Swim) models. But, a study has shown that the comparison was not rigorous since if augmentation are applied on ViT model, than comparison goes like this:

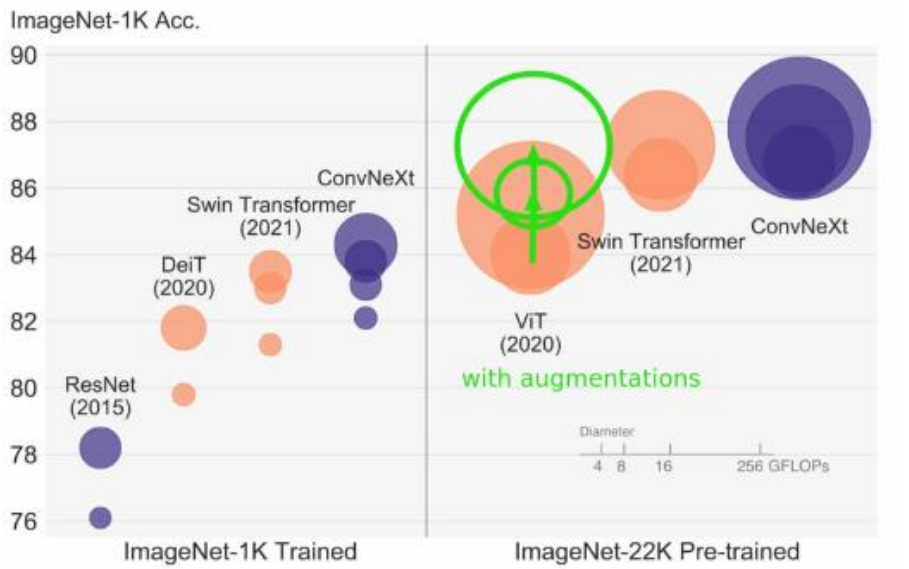

Figure 1. **ImageNet-1K classification** results for • ConvNets and ○ vision Transformers. Each bubble's area is proportional to FLOPs of a variant in a model family. ImageNet-1K/22K models here take $224^2/384^2$ images respectively. We demonstrate that a standard ConvNet model can achieve the same level of scalability as hierarchical vision Transformers while being much simpler in design.

Fig.11 the comparison of ResNet, ViT with ConvNext.

Diameter shows the Computation Power needed, hence, bigger the circle is, more computationally expensive model will be.

Conclusions & Perspectives

Techniques used in this research are not novel. These are techniques that have been used separately in previous research. This research combines these techniques and uses them collectively.

Not only are ConvNeXts competitive with Transformers in solving image classification tasks, but also in solving general-purpose computer vision tasks, i.e., image segmentation and object detection tasks.

The “A ConvNet for the 2020s” paper demonstrates that it is, rather, the many seemingly tiny architecture hyperparameters and not the architecture itself that can tweak the way to state-of-the-art.

References

[1] A Comprehensive Introduction to different Types of Convolutions in Deep Learning.
[2] Visualizing A Neural Machine Translation Model
[3] The Illustrated Transformer
[4] Vision Transformers (ViT) in Image Recognition
[5] A Comprehensive Guide To Swin Transformer
[6] A ConvNet for the 2020s
[7] Swin Transformer: Hierarchical Vision Transformer using Shifted Windows
[8] An Image is Worth 16x16 Words
[9] Deep Residual Learning for Image Recognition
[10] Aggregated Residual Transformations for Deep Neural Networks
[11] Why AdamW matters