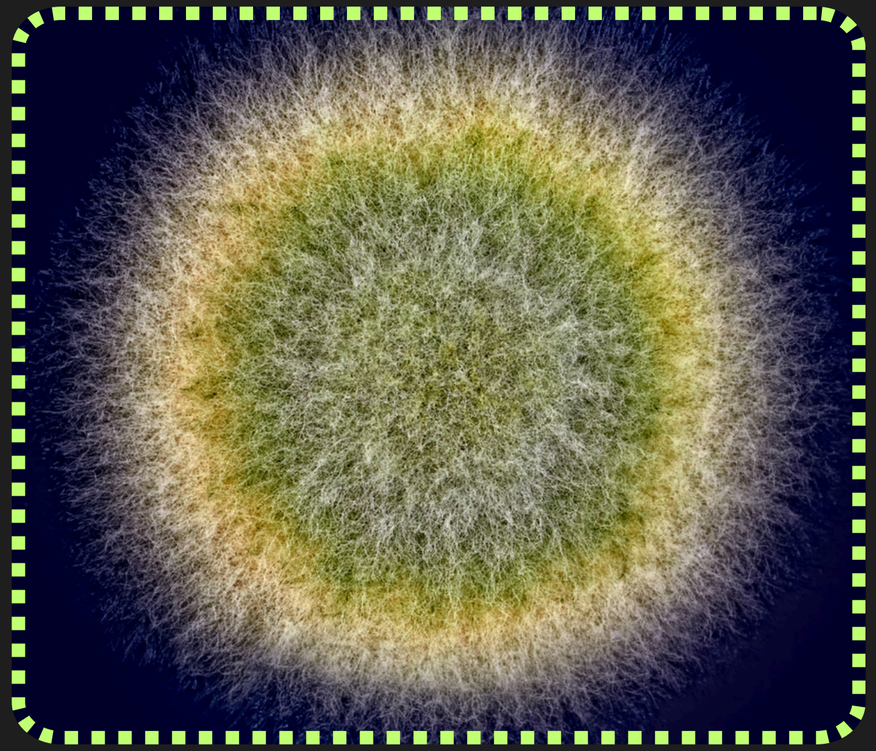
Jules MATHIEU and Tom NORROY

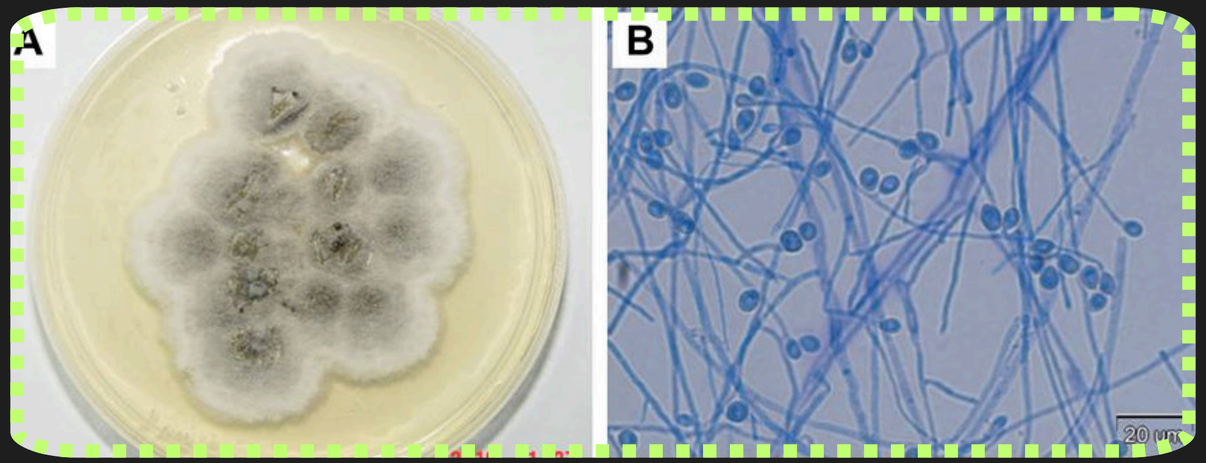
A DEEP CNN ARCHITECTURE : THE RESNET

Application to image classification of pathogenous mycosis

- Fungi are essential both ecologically and industrially, but some species are pathogenic
- Fungal infections cause 1.5 to 2 million deaths per year
- Fungal co-infections in severe COVID-19 cases affect about 10% of patients
- Mucormycosis (“Black Fungi”) surged in India during the pandemic, especially among diabetic patients



- Diagnosis relies on:
  - direct microscopic examination (limited)
  - long culture periods (28–31 days) for definitive identification
- The lack of mycologists in less developed regions limits rapid access to diagnosis
- Computer-Aided Diagnosis (CAD) systems are increasingly used in modern medicine
- An AI-based image classifier could speed up fungal diagnosis and improve patient care.



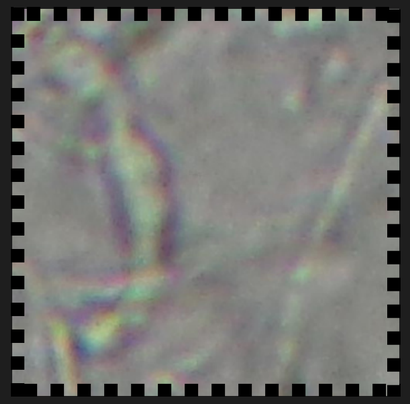
Fungi pre-processed Dataset

- Microscopic images of fungi
- Origin : colombian mycological laboratory
- Original images were pre-processed :
  - Automatic patch extraction (500×500)
  - Artifact removal (dark borders, empty areas, noise)
  - Manual validation with expert mycologists

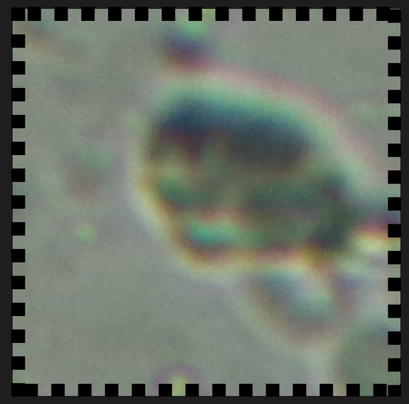
Final dataset :

- 9114 images (500x500 px)
- jpg format
- Class distribution (raw dataset):
  - TSH : 4404 images (H1)
  - BASH : 2324 images (H2)
  - GMA : 819 images (H3)
  - SHC : 818 images (H5)
  - BBH : 739 images (H6)

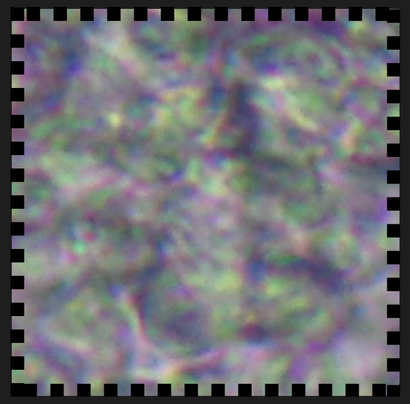
Fungi pre-processed Dataset



H1

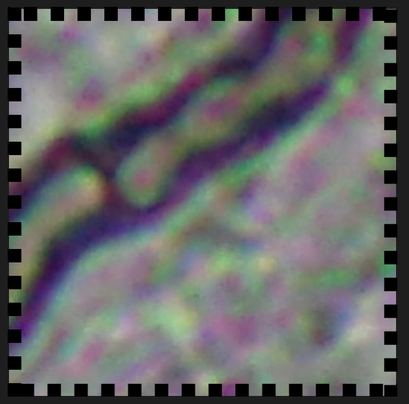


H2



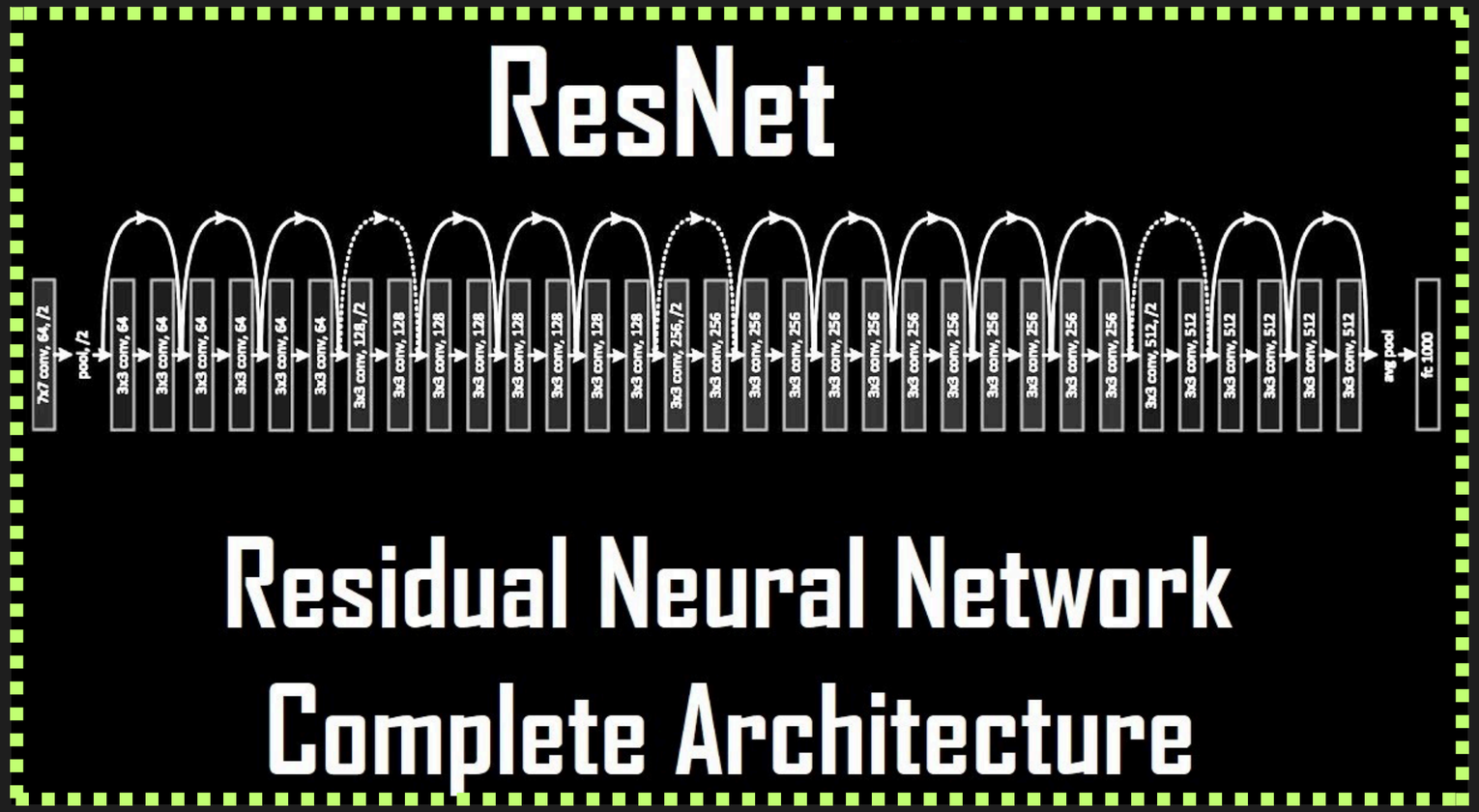
H3

H5



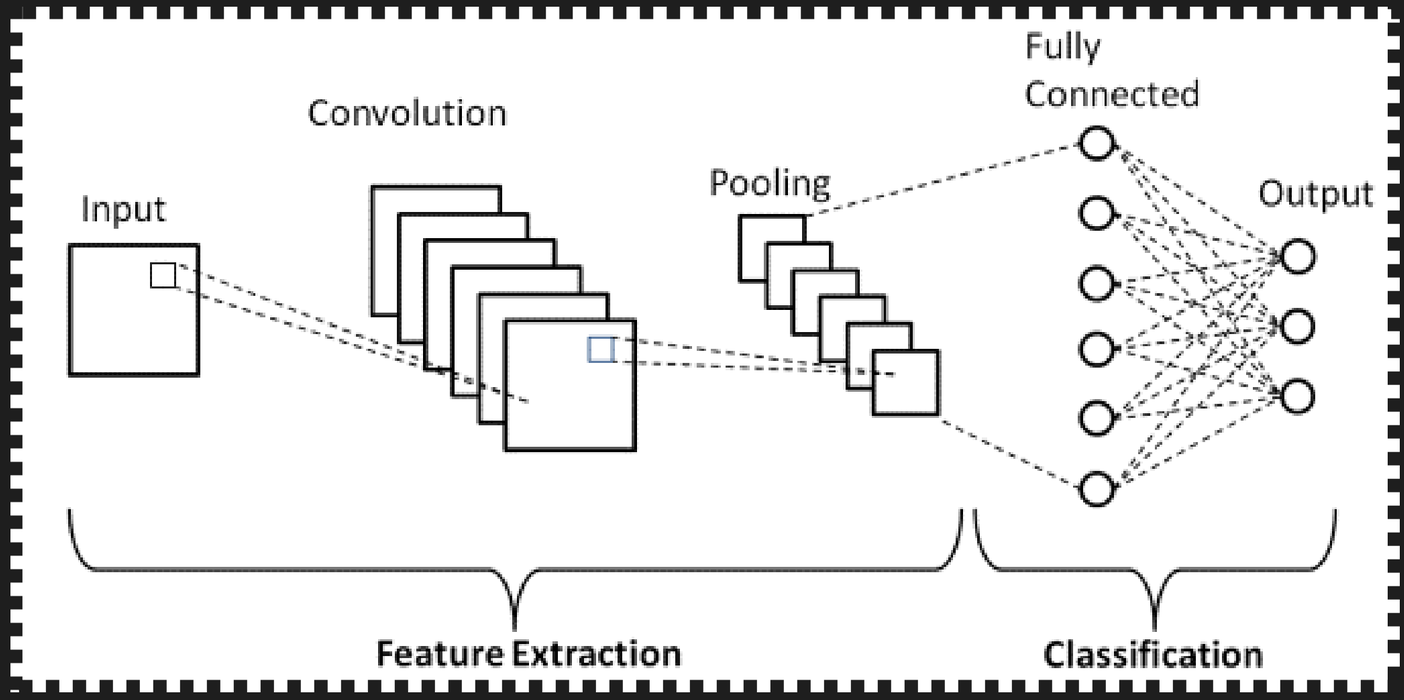
H6

WHAT IS A RESNET ?

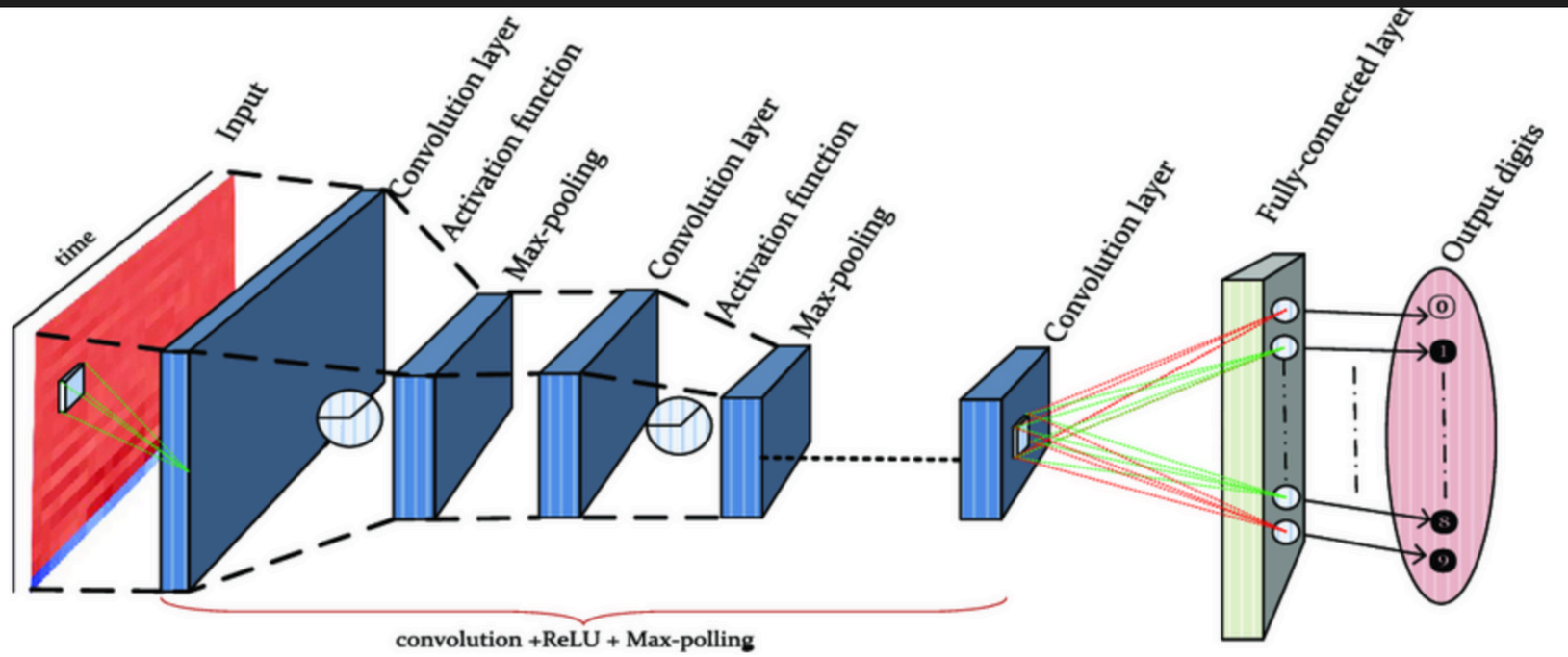


Reminder about Convolutional Neural Networks (CNNs)

- **Convolutional layers** → automatic feature extraction
- **Pooling layers** → dimensionality reduction with key information preserved
- **Fully connected layers** → final classification stage
- High performance in **image classification**, object detection, and medical imaging



Our convolutionnal network



Expliquer les paramètres du CNN

Our convolutionnal network

```
Input (3×224×224)
↓
ConvBlock1 : 32 filtres, MaxPool → 32×112×112
↓
ConvBlock2 : 64 filtres, MaxPool → 64×56×56
↓
ConvBlock3 : 128 filtres, MaxPool → 128×28×28
↓
Flatten → 128×28×28 = 100352 neurones
↓
FC1 (256 neurones) → ReLU → Dropout(0.5)
↓
FC2 (5 classes)
```

- 3 block - compromise complexity/expressivity
- Conv3x3 + padding = 1 - local texture/biblio (VGG)
- [32 → 64 → 128] - growing deepness/texture
- MaxPool - spatial size/speed
- BatchNorm - CNN from scratch/converge/LR ++
- FC - overfitting/features selection

FOCUS : ResNet

Deep Residual Learning for Image Recognition

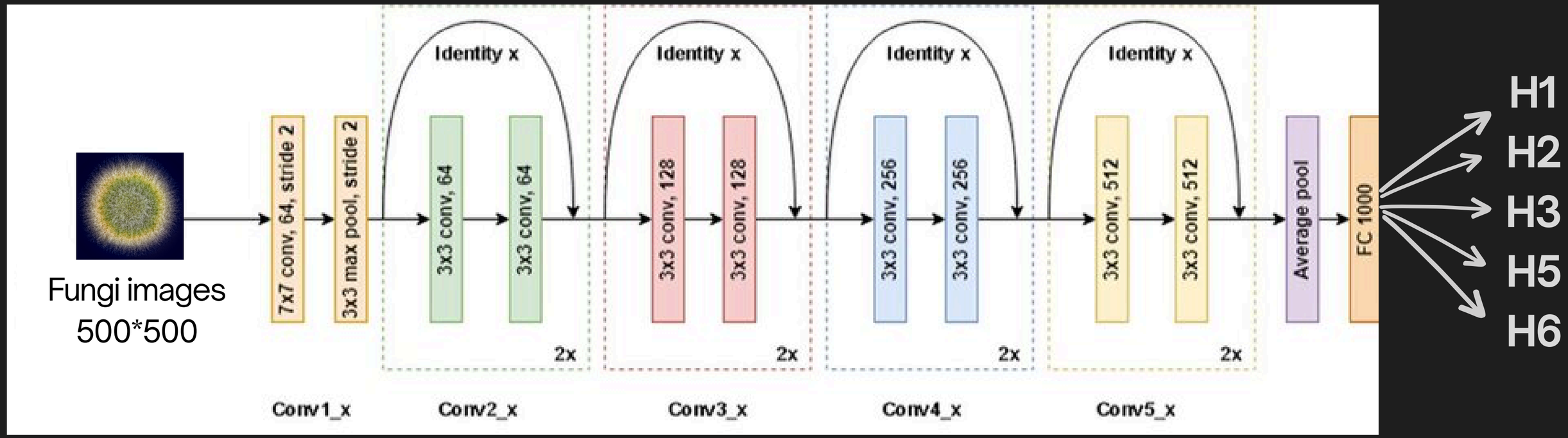
Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun

Microsoft Research

{kahe, v-xiangz, v-shren, jiansun}@microsoft.com

2015

ResNet 18 Architecture



Skip Connections

Goal: facilitate learning by allowing information to pass through without unnecessary distortion

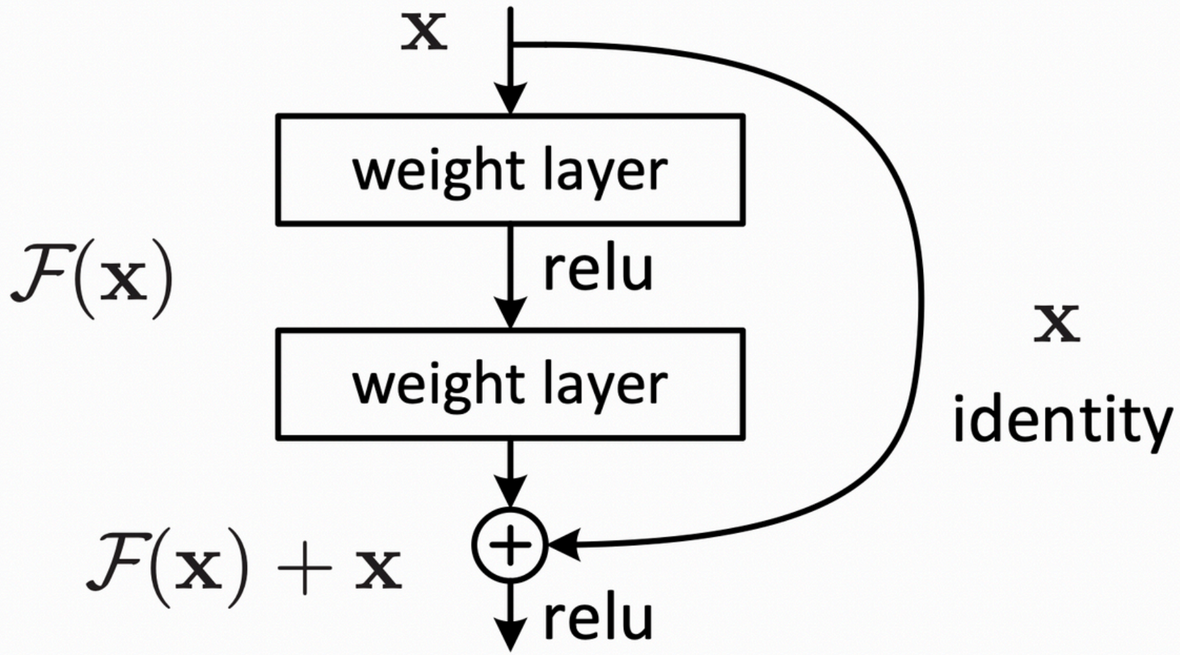


Figure 2. Residual learning: a building block.

Analogy: correcting a text

- Without skip connection → rewrite the entire text at every step
- With skip connection → apply only the needed corrections

Skip Connections

Without skip connection - vanishing gradient

$$x_{l+1} = \mathcal{F}_l(x_l)$$

$$\frac{\partial \mathcal{L}}{\partial x_l} = \frac{\partial \mathcal{L}}{\partial x_{l+1}} \cdot \frac{\partial \mathcal{F}_l}{\partial x_l}$$

$$\boxed{\frac{\partial \mathcal{L}}{\partial x_0} = \frac{\partial \mathcal{L}}{\partial x_L} \cdot \prod_{l=0}^{L-1} \frac{\partial \mathcal{F}_l}{\partial x_l}}$$

With skip connection - transmission

$$x_{l+1} = \mathcal{F}_l(x_l) + x_l$$

$$\frac{\partial \mathcal{L}}{\partial x_l} = \frac{\partial \mathcal{L}}{\partial x_{l+1}} \cdot \left(\frac{\partial \mathcal{F}_l}{\partial x_l} + I \right)$$

$$\boxed{\frac{\partial \mathcal{L}}{\partial x_0} = \frac{\partial \mathcal{L}}{\partial x_L} \cdot \prod_{l=0}^{L-1} \left(\frac{\partial \mathcal{F}_l}{\partial x_l} + I \right)}$$

Skip Connections

- It allows the unaltered input to be added directly to the output of the convolutional layers.

- This bypass connection ensures that essential information from earlier layers is preserved and propagated through the network, even if the convolutional layers struggle to learn additional features in that specific block

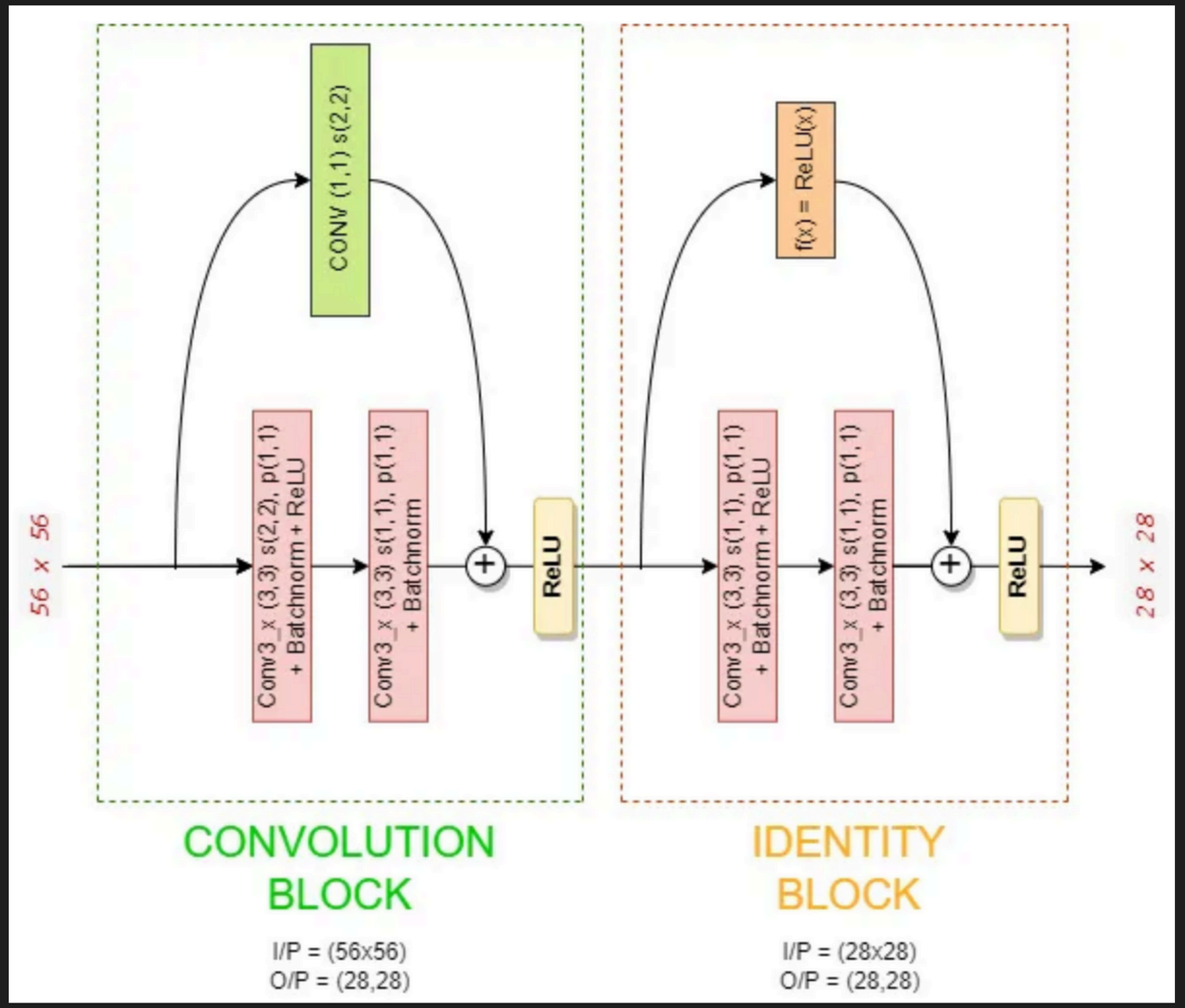
Type of ResNet

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Table 1. Architectures for ImageNet. Building blocks are shown in brackets (see also Fig. 5), with the numbers of blocks stacked. Down-sampling is performed by conv3_1, conv4_1, and conv5_1 with a stride of 2.

Conv3_x block :

- Dimension of the Output and the Input are note the same : Convolution block
- The input and output have the same dimension : Identity block



Conv3_x in pytorch (from scratch)

init :

```
# BLOCK-3 (1) input=(56x56) output = (28x28)
self.conv3_1_1 = nn.Conv2d(in_channels=64, out_channels=128, kernel_size=(3,3), stride=(2,2), padding=(1,1))
self.batchnorm3_1_1 = nn.BatchNorm2d(128)
self.conv3_1_2 = nn.Conv2d(in_channels=128, out_channels=128, kernel_size=(3,3), stride=(1,1), padding=(1,1))
self.batchnorm3_1_2 = nn.BatchNorm2d(128)
self.concat_adjust_3 = nn.Conv2d(in_channels=64, out_channels=128, kernel_size=(1,1), stride=(2,2), padding=(0,0))
self.dropout3_1 = nn.Dropout(p=self.dropout_percentage)
# BLOCK-3 (2)
self.conv3_2_1 = nn.Conv2d(in_channels=128, out_channels=128, kernel_size=(3,3), stride=(1,1), padding=(1,1))
self.batchnorm3_2_1 = nn.BatchNorm2d(128)
self.conv3_2_2 = nn.Conv2d(in_channels=128, out_channels=128, kernel_size=(3,3), stride=(1,1), padding=(1,1))
self.batchnorm3_2_2 = nn.BatchNorm2d(128)
self.dropout3_2 = nn.Dropout(p=self.dropout_percentage)
```

forward :

```
# block3 - 1[Convolution block]
x = self.relu(self.batchnorm3_1_1(self.conv3_1_1(op2)))    # conv3_1
x = self.batchnorm3_1_2(self.conv3_1_2(x))                 # conv3_1
x = self.dropout3_1(x)
# block3 - Adjust
op2 = self.concat_adjust_3(op2) # SKIP CONNECTION
# block3 - Concatenate 1
op3_1 = self.relu(x + op2)
# block3 - 2[Identity Block]
x = self.relu(self.batchnorm3_2_1(self.conv3_2_1(op3_1)))  # conv3_2
x = self.batchnorm3_2_2(self.conv3_2_2(x))                 # conv3_2
x = self.dropout3_2(x)
# op - block3
op3 = self.relu(x + op3_1)
```

ResNet VS Plain

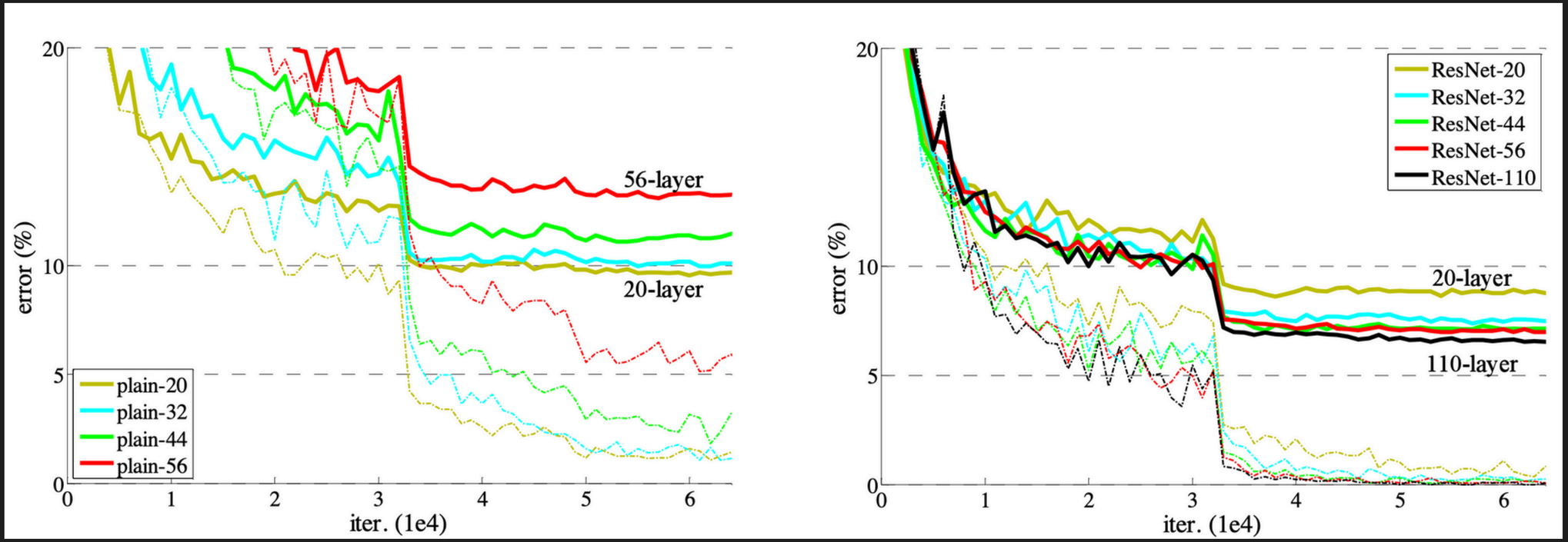
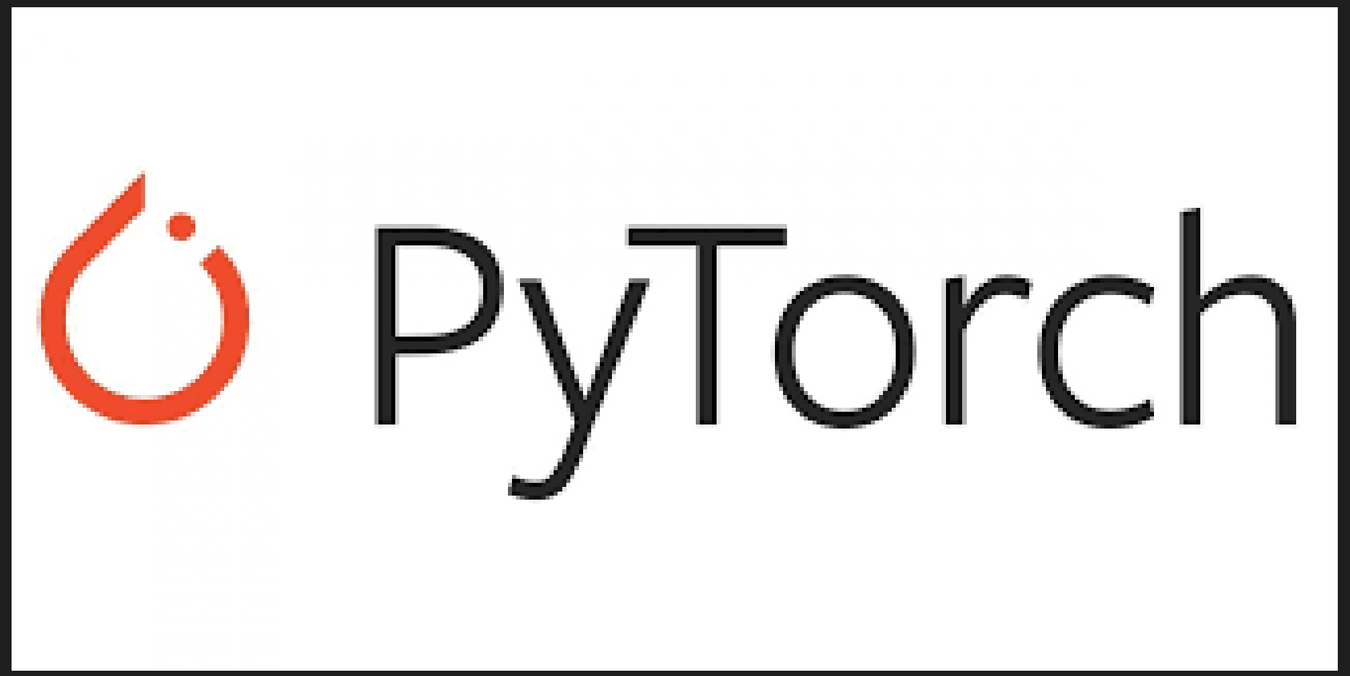


Figure 6. Training on CIFAR-10. Dashed lines denote training error, and bold lines denote testing error.

Left: plain networks. – Right: ResNets.

The error of plain-110 is higher than 60% and not displayed.

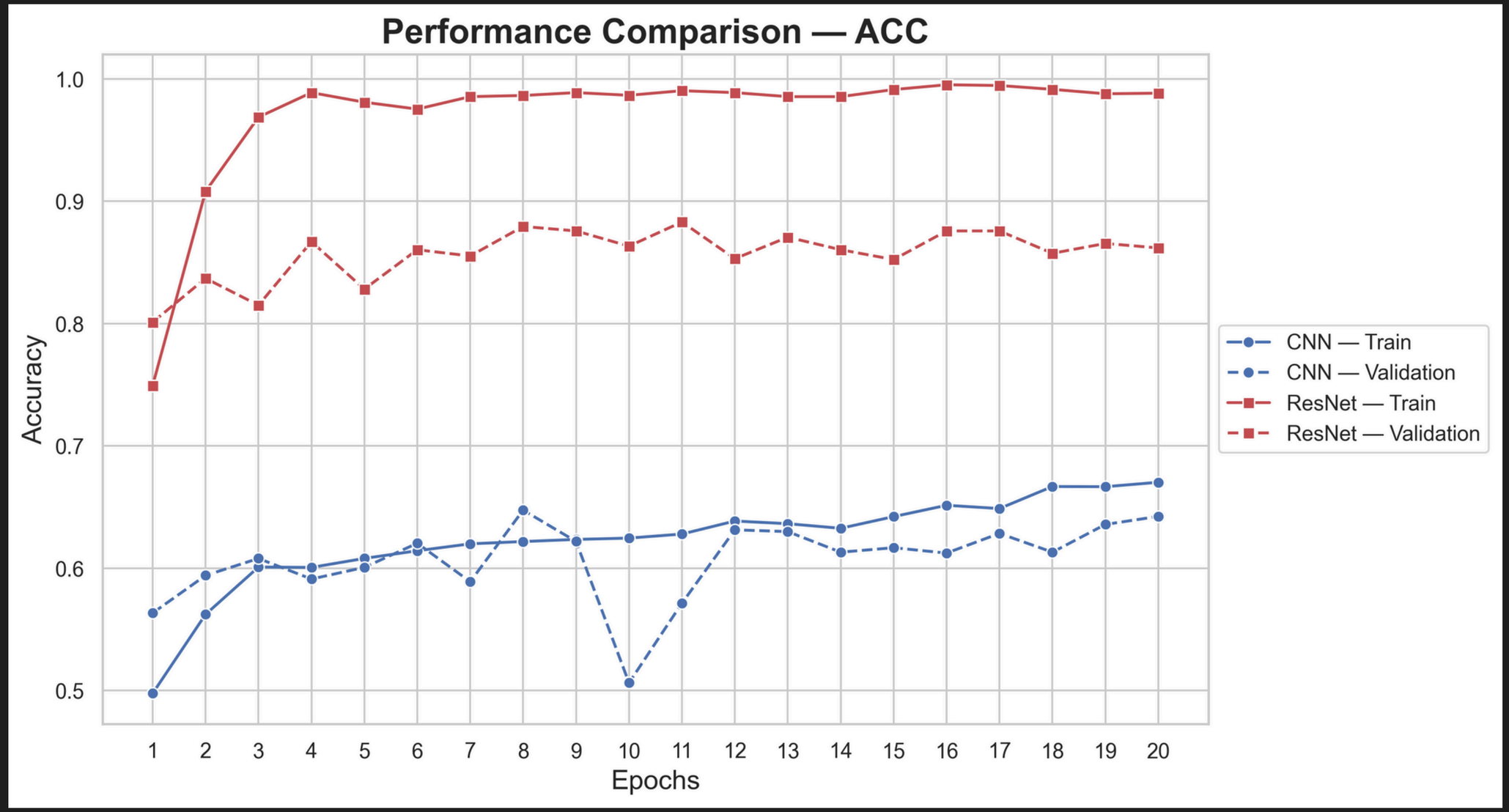
RESNET50 APPLICATION



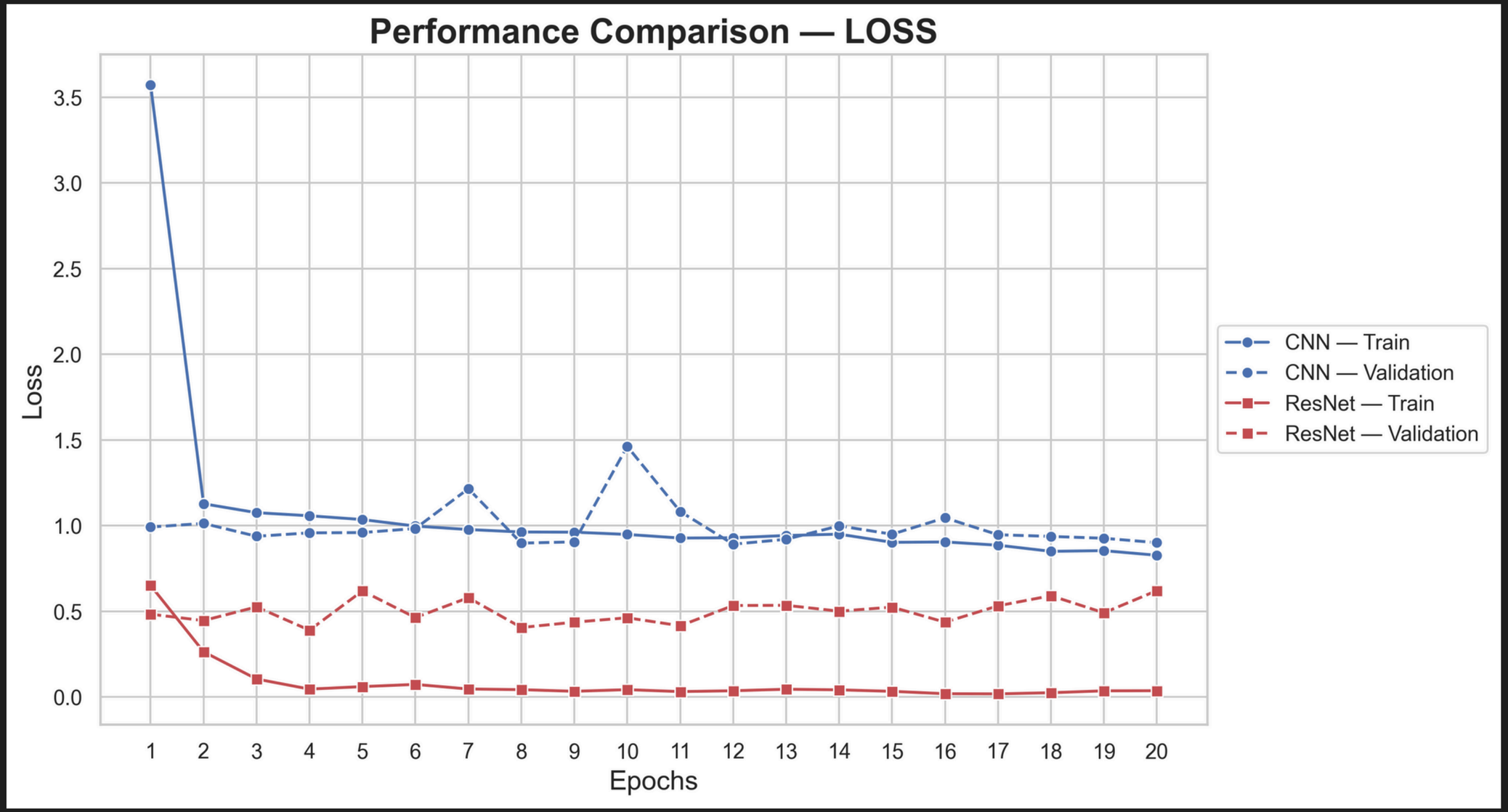
Full ResNet18 used : (Pre-trained with ImageNet)


```
class ResNet18Gray(nn.Module):
    def __init__(self, num_classes):
        super().__init__()

        # Pre-trained model
        self.model = models.resnet18(weights=models.ResNet18_Weights.DEFAULT)

        # Last layer adaptation
        self.model.fc = nn.Linear(512, num_classes)

    def forward(self, x):
        return self.model(x)
```

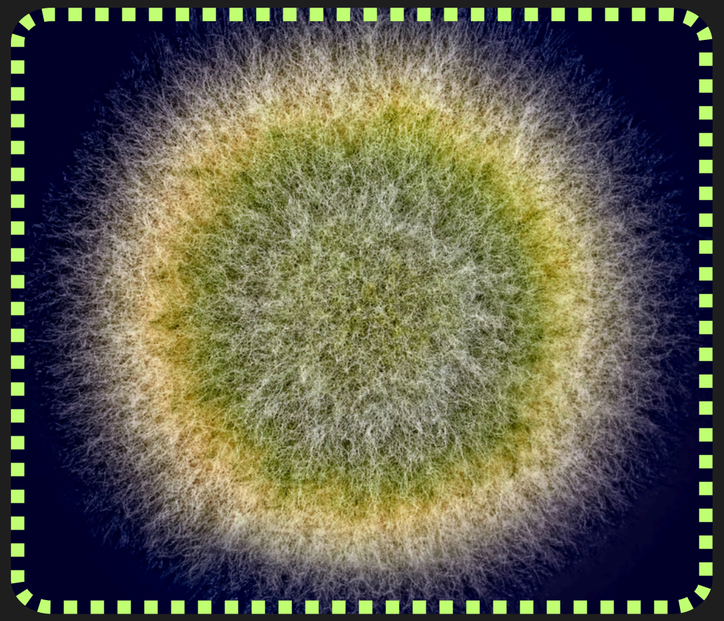
Results on the Fungi Dataset

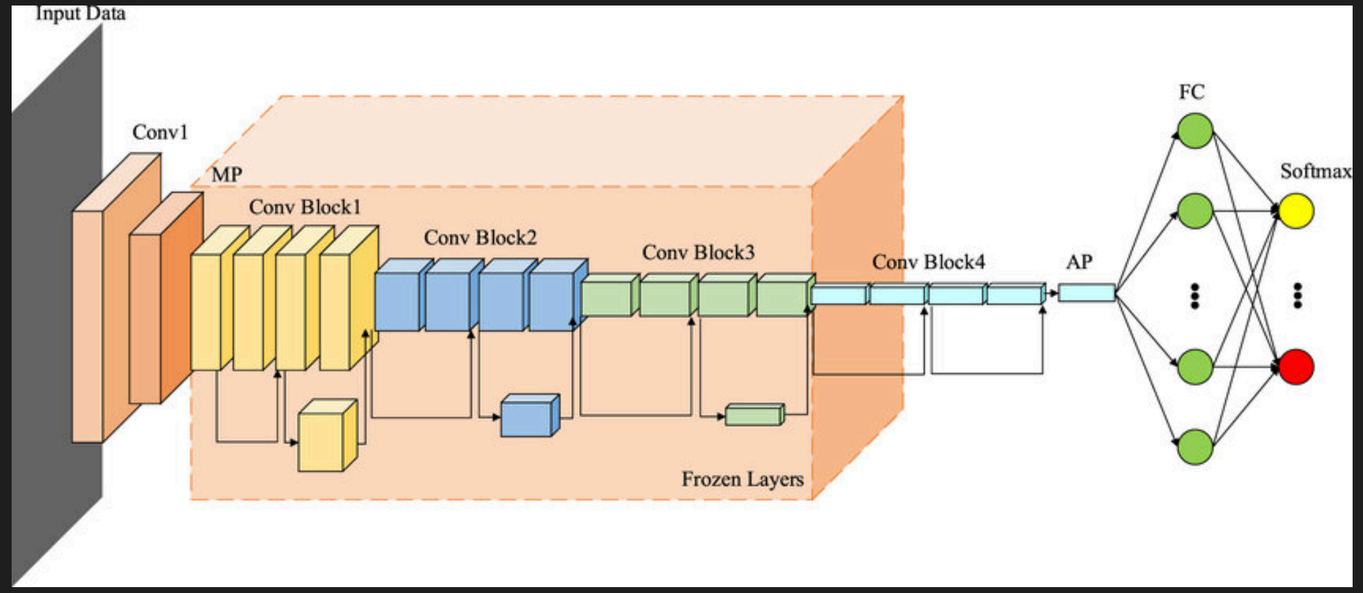


Results on the Fungi Dataset



CONCLUSION




Input Data
Conv1
MP
Conv Block1
Conv Block2
Conv Block3
Conv Block4
AP
FC
Softmax
Frozen Layers

DeFungi: Direct Mycological Examination of Microscopic Fungi Images

Camilo Javier Pineda Sopo, Farshid Hajati, Soheila Gheisari

- The study classifies the five types of fungi using deep learning
- Three CNN models are evaluated: VGG16, Inception V3, and **ResNet50**
- Two approaches are compared:
  - Models trained from scratch
  - **Pre-trained** models using the ImageNet dataset
- The goal is to benchmark and compare classification performance across these approaches

Table 1: 10-fold test accuracy performance comparison of the models using transfer learning.

ResNet50			VGG16			Inception V3		
Fold	Loss	Accuracy	Fold	Loss	Accuracy	Fold	Loss	Accuracy
1	0.894	85.199%	1	0.604	84.799%	1	0.815	79.600%
2	0.968	83.600%	2	0.674	82.400%	2	0.399	87.999%
3	0.865	85.600%	3	0.447	88.800%	3	0.757	80.000%
4	0.715	80.000%	4	0.523	85.600%	4	0.762	83.600%
5	0.814	82.800%	5	0.68	83.999%	5	0.742	76.399%
6	1.118	83.999%	6	0.573	83.200%	6	0.64	82.800%
7	1.099	80.000%	7	0.498	86.799%	7	0.719	84.799%
8	0.68	83.999%	8	0.76	84.399%	8	0.946	81.999%
9	0.692	84.399%	9	0.748	83.600%	9	0.763	82.400%
10	0.927	80.800%	10	0.44	86.799%	10	0.406	88.400%
Average Loss:		0.877	Average Loss:		0.5947	Average Loss:		0.6949
Average Accuracy:		**83.040%**	Average Accuracy:		**85.040%**	Average Accuracy:		**82.800%**
Standard Deviation:		1.969%	Standard Deviation:		1.861%	Standard Deviation:		3.505%

THANK YOU FOR YOUR ATTENTION