RESNET (RESNET50, RESNET101, RESNET152) SUMMARY

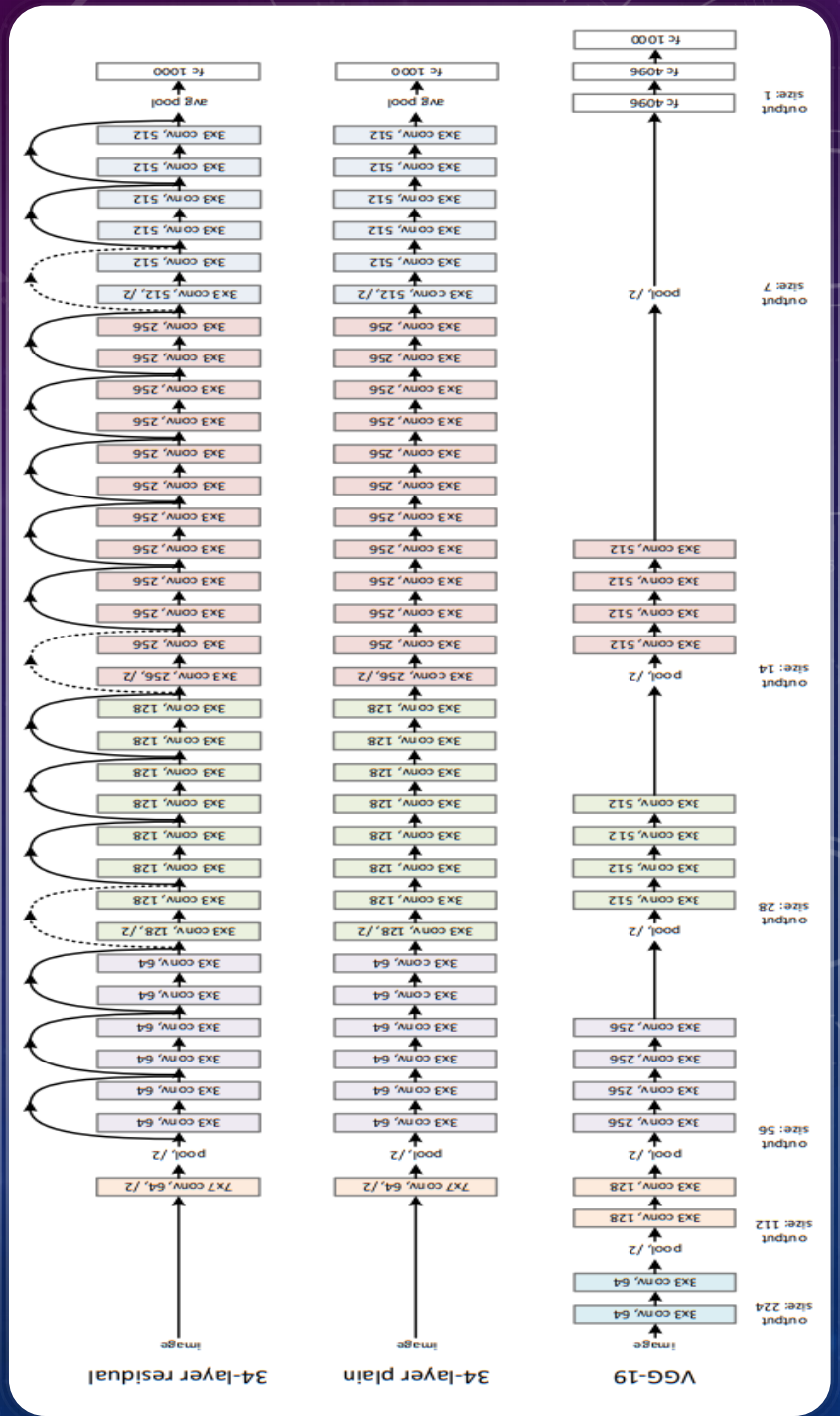
HOW IT WORKS, APPLICATIONS IN IMAGE CAPTIONING, AND PERFORMANCE

WHAT IS RESNET

A residual neural network (also referred to as a residual network or ResNet)[1] is a deep learning architecture in which the layers learn residual functions with reference to the layer inputs. It was developed in 2015 for image recognition, and won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) of that year.

As a point of terminology, "residual connection" refers to the specific architectural motif of
$x \mapsto f(x)+x$
where f is an arbitrary neural network module. The motif had been used previously (see §History for details). However, the publication of ResNet made it widely popular for feedforward networks, appearing in neural networks that are seemingly unrelated to ResNet

WHAT IS RESNET STRUCTURE



HOW RESNET WORKS

- ResNet (Residual Network) is a CNN designed to solve the vanishing gradient problem.
- Uses Residual Connections (Skip Connections) to retain information across layers.
- Composed of multiple Residual Blocks with Convolution, Batch Normalization, and ReLU Activation.
- Allows training of deep networks without performance degradation.

RESNET VERSIONS & LAYER COUNT

- - ResNet50 → 50 layers
- - ResNet101 → 101 layers
- - ResNet152 → 152 layers (deepest and most accurate)

1. HOW DOES RESNET WORK?

📌 **Key Components of ResNet:**
✅ **Convolutional Layers** for feature extraction from images.
✅ **Batch Normalization** to speed up training and improve stability.
✅ **ReLU Activation** for non-linearity and better learning.
✅ **Skip Connections** to retain original information across layers.

📌 **Popular ResNet Versions and Their Layer Counts:**

- **ResNet50** → contains **50 layers**.
- **ResNet101** → contains **101 layers**.
- **ResNet152** → contains **152 layers** (deeper and more accurate).

RESNET IN IMAGE CAPTIONING

- Used as a Feature Extractor before passing to LSTM or Transformer.
- Converts an image into numerical feature vectors.
- Steps: Load ResNet, extract features, feed into LSTM/Transformer to generate text.

CODE EXAMPLE (FEATURE EXTRACTION)

```
import torch
import torchvision.models as models
import torchvision.transforms as transforms
from PIL import Image

# Load ResNet50 without the final layer
resnet = models.resnet50(pretrained=True)
resnet = torch.nn.Sequential(*list(resnet.children())[:-1])

# Transform Image
transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
])
```

RESOURCE CONSUMPTION

- ResNet50 is the fastest and most lightweight.
- ResNet152 is more accurate but requires more resources.
- Can be optimized using Mixed Precision Training and TensorRT.

ACCURACY & EVALUATION

- ResNet Accuracy (ImageNet):
- ResNet50: 76.2% (Top-1), 92.8% (Top-5)
- ResNet101: 77.4% (Top-1), 93.6% (Top-5)
- ResNet152: 78.6% (Top-1), 94.2% (Top-5)

- For Image Captioning, measure performance using BLEU, CIDEr, METEOR scores.

BLEU SCORE EXAMPLE

```
from nltk.translate.bleu_score import sentence_bleu

# Reference & Generated Sentences

reference = [['a', 'cat', 'is', 'sitting', 'on', 'a', 'mat']]

candidate = ['a', 'cat', 'sits', 'on', 'the', 'mat']

# Compute BLEU Score

core = sentence_bleu(reference, candidate)

print(f'BLEU Score: {score:.2f}')
```

COMPARISON BETWEEN OTHER MODELS

Architecture	Model Variant	Parameters	ImageNet Top-1 Accuracy	FLOPs	Approx. Inference Time
VGG	VGG16	~138M	~71.5%	~15.3B	Slow (high compute & memory use)
	VGG19	~144M	~71.3%	~19.6B	Slow (even heavier than VGG16)
ResNet	ResNet50	~25.6M	~76.2%	~4.1B	Moderate; well-optimized for deep learning
	ResNet101	~44.5M	~77.4%	~7.6B	Moderately slower than ResNet50
	ResNet152	~60.2M	~78.6%	~11.3B	Slower, more compute-intensive
Inception	GoogLeNet (Inception v1)	~6.8M	~68.3%	~1.5B	Fast and lightweight
	InceptionV3	~23.8M	~77.9%	~5B	Relatively fast with good accuracy
	InceptionV4	~42M	~80.0% (approx.)	~13B	Slower than V3, higher accuracy
EfficientNet	EfficientNet-B0	~5.3M	~77.1%	~0.39B	Very fast & efficient
	EfficientNet-B7	~66M	~84.3%	~37B	Slower than B0; heavy but highest accuracy
DenseNet	DenseNet121	~8M	~74.9%	~2.8B	Moderate; efficient parameter usage
	DenseNet169	~14M	~76.2%	~3.4B	Slightly slower than DenseNet121
	DenseNet201	~20M	~77.3%	~4.4B	Similar trends; competitive speed

SOME RESOURCES

https://youtu.be/WDwNem7kiCU?si=IQ357_akqRahkviD

Image Captioning using ResNet

sanandita001/Image-captioning: Image Captioning Resnet 50 and LSTM and Flickr8k

Image Caption->ResNet,TransformerDecoder[PyTorch]

Image Captioning using ResNet and LSTM

https://youtu.be/FJr5DZpbx3A?si=bFyRYmup66UKQvhP

🎯 SUMMARY

- ResNet uses Residual Connections to enable deep training.
- Used in Image Captioning as a feature extractor.
- ResNet50 is the best for balance, ResNet152 is more accurate but slower.
- Accuracy: 76%-78% in classification, measured using BLEU, CIDEr, METEOR in captioning.