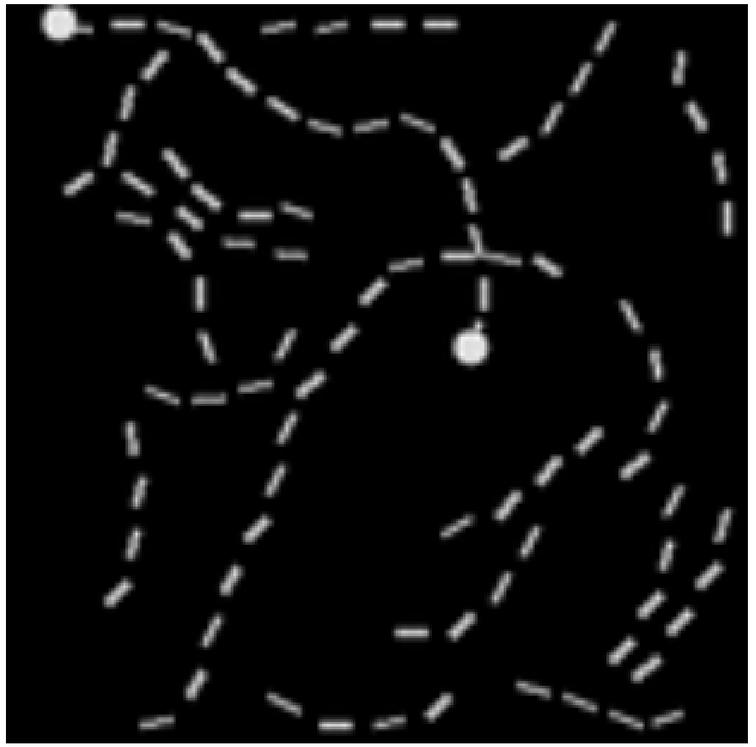
2024

Long-range-arena Benchmarking on Pathfinder dataset

CHOUKRI Zakaria
ARNAOUI Basma
AIT MAGOURT Adnane

Supervised By:
Hamza ALAMI
Issam AIT YAHIA

Pathfinder Dataset



- Images used: 32x32 pixels
- Binary task: Identify if two points are connected by a dashed path.
- We used images from curv_contour_length_14: Includes images with long contours of length 14, presenting the most challenging task.
- Total of 200.000 images
- We split the dataset into training, validation, and testing sets (80%, 10%, 10% split).

Models evaluated

CHOUKRI Zakaria

Model	# Parameters	Accuracy	Training time	Efficiency
CNN Model	2226434	0.7968	3193.16 s	0.0068
DEIT Model	21666434	0.7883	10064.90 s	0.0051
Transformer model	2631490	0.5011	10138.11 s	0.0037

Models evaluated

AIT MAGOURT Adnane

Model	# Parameters	Accuracy	Training time	Efficiency
CNN model	543490	0.5009	1291.06 s	0.0053
ViT model	1983650	0.5005	1428.65 s	0.0048
Transformer	3359426	0.5017	2926.44 s	0.0042

CNN Model

- Extends pytorch_lightning.LightningModule to define a CNN-based classification model with the following architecture:
  - Convolutional layers: Two blocks of convolution, ReLU activation, and max pooling to extract hierarchical spatial features.
  - Fully connected layers: A classifier with two dense layers and a final output size of 2 (binary classification).
  - Flattening reshapes the tensor to feed the linear layers after the convolutional feature extraction.
- Input: 1×32×32 grayscale images.
- Specifies the optimizer (Adam) with a learning rate of 10^-3
- uses Cross-Entropy Loss

Vision Transformer (ViT) Model

- Uses the ViT class from the vit_pytorch library.
- divides the input image into patches, embeds each patch, and processes them using a transformer encoder. Finally, it performs classification using a linear layer.
- image_size=32: The input image dimensions.
- patch_size=4: Divides each 32×32 image into 8×8=64 patches, each of size 4×4.
- num_classes=2: Binary classification output.
- dim=128: Embedding dimension for the patches.
- depth=6: Number of transformer encoder layers.
- heads=8: Multi-head self-attention with 8 attention heads.
- mlp_dim=256: Feed-forward layer dimension inside each transformer encoder block.
- channels=1: For grayscale input images.

Transformer-based model

Feature Extraction

- Three convolutional layers reduce the spatial dimensions of the input image while increasing feature depth.
- Batch normalization and ReLU activations are used after each convolution to stabilize and enhance training.

Positional Encoding

- A learnable sinusoidal positional encoding is added to the feature maps to preserve positional information.
- Ensures that the Transformer can understand the sequence and spatial structure of features.

Transformer-based model

- **Transformer Encoder**:
- Built using nn.TransformerEncoder with num_layers=4.
- Each layer consists of multi-head attention and feedforward sub-layers.
- Dropout regularization (0.1) is applied to prevent overfitting.
- **Classification Head**
- The encoded features are pooled globally using mean pooling.
- A fully connected network reduces the features to two output logits (binary classification).
- Loss: Cross-entropy loss is used for binary classification.

Models evaluated

ARNAOUI Basma

Model	# Parameters	Accuracy	Training time	Efficiency
Vit Model	85800194	0.85135000944	13697.67	0.0049
Mobile Vit Model	191298	0.4975	0.0046	0.0046
EfficientFormer Model	11393724	0.9727	5107.9s	0.0072
Swin Transformer	86745274	0.5023	16384.67	0.0028

Vision Transformer (Vit) Model

- **Data Preparation for Vit:** to prepare images for the Vision Transformer (ViT) by ensuring uniform size, format, and normalization.

-> Image Resizing:All images resized to 224x224 for compatibility with the model.
-> Normalization: xn = (x-mean)/std =>ensures that all pixel values contribute equally to the learning process and align with the distribution expected by pretrained models, preventing input mismatches.
-> Conversion to Tensors: Transformed images into tensors.

- **Data Splitting + Error Handling**

Vision Transformer (Vit) Model

Vision Transformer (Vit) Model

- **Model Name:** Vision Transformer (ViT)
- **Pretrained On:** ImageNet dataset (1 million+ images, 1,000 classes)
- **Architecture:** Utilizes Transformer-based attention for image classification instead of traditional convolutional neural networks (CNNs).
- **How I Used It:**

Loaded a pretrained ViT model (vit_base_patch16_224) via timm.
Fine-tuned the model for binary classification by modifying the classification head.
Used PyTorch Lightning for efficient training and evaluation.

- **What It Does:**

Splits images into patches and converts them into tokens.
Processes these tokens using self-attention mechanisms to learn relationships between patches.
Outputs a classification prediction, distinguishing between the two classes in my dataset.

- **Why :** State-of-the-art attention mechanism ensures effective feature extraction for image classification tasks.

Mobile Vit Model:A Fusion of Convolution and Transformer

- **What is it:** A lightweight hybrid model combining Convolutional Layers and Transformers.
-
- **How it works:**

->Convolutional Stem: Extracts low-level features.
->MobileViT Block: Processes abstract features with convolutions.
->Transformer Encoder: Captures global dependencies using self-attention.
->Classification Head: Predicts class labels from global feature vector.

- **Why:** Efficient for large datasets (low memory and computational requirements)

Initialized with standard distributions to enhance performance.

Mobile Vit Model: common questions

- **Why use a Transformer Encoder?**

->Transformers enable the model to learn long-range dependencies and relationships between patches, which CNNs might struggle with.(cat example)

- **Why use Convolutions before the Transformer?**

->Convolutions preprocess the image into smaller feature maps, reducing computational cost for the Transformer.

- **What does d_model, nhead, and dim_feedforward mean?**

->d_model: Dimensionality of the feature vectors passed to the Transformer.
->nhead: Number of attention heads in the multi-head attention mechanism.
->dim_feedforward: Dimensionality of the intermediate layer in the Transformer feedforward network.

- **Why use Global Average Pooling?**

->Reduces the feature map into a single feature vector, ensuring the output is independent of input size and focuses on global features.

EfficientFormer Model

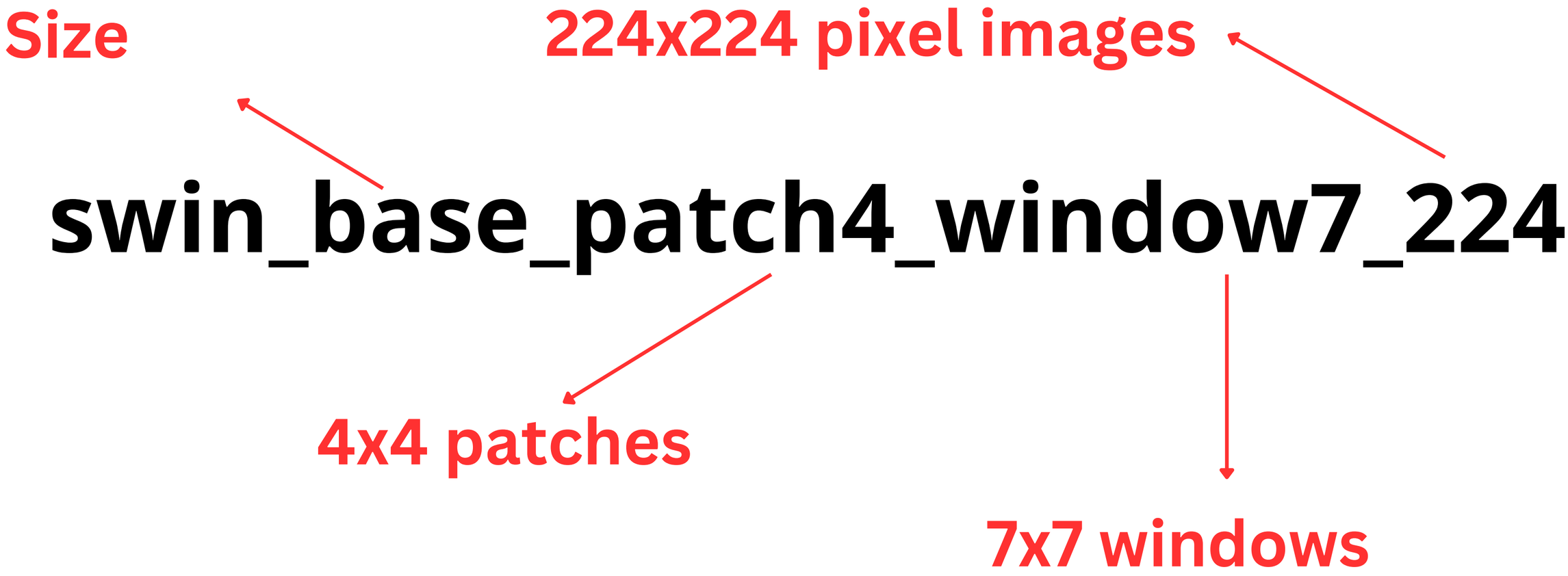
- **What is it :** A transformer-based model optimized for computational efficiency and high accuracy, pretrained on the ImageNet dataset for generalizable feature extraction.

- **How it works:**

->Patch Embedding: Converts images into smaller, processable units (patches), that are then flattened to get patch tokens.
->Transformer Encoder: Captures global dependencies using multi-head self-attention.
->Classification Head: Predicts class probabilities for binary classification.

- **Why:** Combines high accuracy with low computational requirements, pretrained knowledge enhances feature extraction.

Shifted Window (Swin) Transformer



Swin Transformer Model

- **Architecture :** Employs Shifted Window Attention for efficient local and global feature extraction.
- **How I Used It :**

->Loaded the pretrained Swin Transformer model via timm.
->Fine-tuned the model for binary classification by modifying the classification head.
->Used PyTorch Lightning for streamlined training and evaluation.

- **What It Does :**

->Splits images into patches (4×4) and encodes them as tokens.
->Processes tokens with Shifted Window Attention to learn both local and global relationships.
->Builds a hierarchical representation of the image by merging tokens at each stage.
->Outputs predictions, distinguishing between two classes in my dataset.

- **Why I Used It:**State-of-the-art attention mechanism ensures precise feature extraction.

Parameters and Configurations Across Models

AdamW Optimizer:

- Combines the adaptive learning rate of Adam with weight decay regularization, improving generalization.

Loss Function : Cross Entropy Loss

- Standard for classification tasks.
- Measures the discrepancy between predicted class probabilities and true class labels.

Training Framework: Pytorch Lightning

- Unified framework for efficient training, validation, and testing.

Data Augmentation:

- Resize (224×224), Normalize, Convert to Tensor.

Activation function : GeLU

Conclusion

While CNNs demonstrate simplicity and moderate performance, transformer-based models such as EfficientFormer stand out by achieving the highest accuracy with relatively low computational requirements, showcasing their capability to handle long-range dependencies effectively