CHAE: CNN Hybridized with Attention Encoders

Chiwoo Jeon

December 2024

Abstract

In recent years, Transformer-based models have gained significant attention in the field of vision deep learning due to their remarkable performance across various tasks, such as VIT(Vision Transformer) Dosovitskiy et al. [2021] . However, these models are often dependent on vast amounts of training data and computational resources, making them less feasible in environments with limited data availability. To address these challenges, this study introduces the chae model, which stands for CNN Hybridized Attention Encoder.

The chae model is designed with a modest parameter only count of approximately 400,000, yet it demonstrates impressive performance even with a dataset of only 4,000 samples with task of Artwork-Artist Classification. It achieves an accuracy of approximately 85% on the test dataset, showcasing its ability to learn and generalize effectively in low-data scenarios. This result contrasts with the common perception that Transformer models require large-scale datasets for effective training, emphasizing the efficiency and robustness of the chae model.

Furthermore, based on its success with small-scale datasets, the chae model is expected to perform well on more complex and extensive datasets, such as ImageNet. This potential highlights its adaptability and scalability for practical applications where data resources are constrained. In this paper, I provide a detailed analysis of the chae model' s architecture, training methodology, and performance evaluation across various datasets, demonstrating its effectiveness and broad applicability.

Model Inspiration and Architecture

This model is inspired by the human visual circuit. Human visual perception reflects the colors, features, and spatial attributes of objects and processes multiple objects and images simultaneously to understand context. I propose a novel architecture that mimics the structure of the human optic nerve by capturing the visual features of an image and using them to infer contextual information.

Visual Feature Mapping Using CNN

Specifically, the model utilizes CNN layers to map the visual features of an image onto an output feature map. For instance, if the final CNN output has a spatial resolution of 16×16 with 128 feature channels, this visual representation can be interpreted as the activation levels of 128 visual components at each position in the 16×16 spatial grid.

Transformer Integration with Visual Tokens

The CNN output is reshaped into a tensor of size 256×128 (representing 256 spatial positions, each with 128 features) and fed into the input layer of a transformer. The transformer treats the 128-dimensional information at each spatial position as a single token. The following steps outline this process:

1. Each spatial position in the 16×16 grid is flattened, resulting in 256 tokens.
2. Each token is represented by a 128-dimensional vector corresponding to the visual features at that position.
3. The transformer performs attention operations across all 256 tokens, capturing the relationships between the visual elements in the image.

This design enables the model to efficiently capture and contextualize the relationships between different visual components, emulating the way the human optic nerve processes spatial and contextual information from visual stimuli.

CHAE Model Architecture

The **CHAE** model integrates convolutional neural networks (CNN) with transformer-based attention mechanisms to effectively process image data, even with limited training samples. The architecture can be described mathematically as follows:

1. Input Layer

Let the input image be represented as:

$$\mathbf{X} \in R^{H \times W \times C}$$

where $H = 256$, $W = 256$, and $C = 3$ denote the height, width, and number of color channels, respectively.

2. Convolutional Feature Extraction

The input undergoes a series of convolutional layers with batch normalization, activation, pooling, and spatial dropout (with dropout rate $\gamma = 0$) I also adopted model sequences which is differ from conventional models introduced in previous studies such as Resnet He et al. [2015], VGGnet Simonyan and Zisserman [2015]. I put bs norm layer before the convolutional layer, inspired by pre-normalization in transformer model introduced in Xiong et al. [2020].:

$$\begin{aligned}
\mathbf{X}_1 &= \text{BatchNorm}(\mathbf{X}) \\
\mathbf{X}_2 &= \text{GELU}\left(\text{Conv2D}_{32}(\mathbf{X}_1)\right) \\
\mathbf{X}_3 &= \text{MaxPool}(\mathbf{X}_2) \\
\mathbf{X}_4 &= \text{SpatialDropout}(\gamma, \mathbf{X}_3) \\
&\vdots \\
\mathbf{X}_n &= \text{Conv2D}_{128}(\mathbf{X}_{n-1}) \\
\mathbf{X}_{n+1} &= \text{GELU}(\mathbf{X}_n) \\
\mathbf{X}_{n+2} &= \text{MaxPool}(\mathbf{X}_{n+1}) \\
\mathbf{X}_{n+3} &= \text{SpatialDropout}(\gamma, \mathbf{X}_{n+2})
\end{aligned}$$

After the convolutional blocks, the feature map has dimensions:

$$\mathbf{X}_{\text{CNN}} \in R^{16 \times 16 \times 128}$$

3. Patch Encoding with CLS Token

The feature map is reshaped into patches and augmented with a classification (CLS) token:

$$\mathbf{X}_{\text{reshaped}} = \text{Reshape}(\mathbf{X}_{\text{CNN}}, (256, 128))$$

where $256 = 16 \times 16$ represents the number of patches, and 128 is the projection dimension.

A learnable CLS token $\mathbf{CLS} \in R^{1 \times 128}$ is prepended to the patch embeddings:

$$\mathbf{X}_{\text{cls}} = \mathbf{CLS} \oplus \mathbf{X}_{\text{reshaped}} \in R^{257 \times 128}$$

where $\oplus$ denotes concatenation.

Position embeddings $\mathbf{E}_{\text{pos}} \in R^{257 \times 128}$ are added to incorporate positional information:

$$\mathbf{X}_{\text{pos}} = \mathbf{X}_{\text{cls}} + \mathbf{E}_{\text{pos}}$$

4. Transformer Encoder Layers

The original Transformer model, proposed by Vaswani et al. [2023] in *Attention Is All You Need*, introduced self-attention mechanisms and positional encoding, allowing for greater parallelization and scalability in machine translation tasks. The transformer component consists of $L = 2$ identical layers, each comprising multi-head self-attention and a multilayer perceptron (MLP) with residual connections and layer normalization.

Multi-Head Self-Attention

For each transformer layer $l = 1, 2$:

$$\mathbf{Z}_l = \text{LayerNorm}(\mathbf{X}_{\text{pos}})\mathbf{A}_l = \text{MultiHeadAttention}(\mathbf{Z}_l, \mathbf{Z}_l, \mathbf{Z}_l)\mathbf{X}_l = \mathbf{A}_l + \mathbf{X}_{\text{pos}}$$

Feed-Forward Network (MLP)

$$\mathbf{M}_l = \text{LayerNorm}(\mathbf{X}_l)\mathbf{F}_l = \text{Dropout}(\text{GELU}(\mathbf{M}_l\mathbf{W}_1 + \mathbf{b}_1))\mathbf{W}_2 + \mathbf{b}_2\mathbf{X}_{l+1} = \mathbf{F}_l + \mathbf{X}_l$$

where $\mathbf{W}_1 \in R^{128\times 256}$, $\mathbf{W}_2 \in R^{256\times 128}$, and $\mathbf{b}_1, \mathbf{b}_2$ are bias terms.

After L layers, the transformer output is:

$$\mathbf{X}_{\text{transformer}} = \mathbf{X}_L \in R^{257\times 128}$$

5. Classification Head

The CLS token representation is extracted and passed through layer normalization and a dense layer with softmax activation for classification:

$$\mathbf{X}_{\text{CLS}} = \mathbf{X}_{\text{transformer}}[0] \in R^{128}\mathbf{X}_{\text{norm}} = \text{LayerNorm}(\mathbf{X}_{\text{CLS}})\mathbf{Y} = \text{Softmax}(\mathbf{X}_{\text{norm}}\mathbf{W}_c + \mathbf{b}_c) \in R^C$$

where $\mathbf{W}_c \in R^{128\times C}$ and $C = 11$ is the number of classes.

6. Model Summary

Combining all components, the **CHAE** model can be succinctly represented as:

$$\mathbf{Y} = \text{Softmax}\left(\text{LayerNorm}\left(\text{CLS}\left(\text{Transformer}\left(\text{PatchEncoder}\left(\text{CNN}(\mathbf{X})\right)\right)\right)\right)\mathbf{W}_c + \mathbf{b}_c\right)$$

Key Parameters

- **Convolutional Layers:** Four Conv2D layers with filter sizes 32, 32, 64, and 128.
- **Projection Dimension:** $D = 128$.
- **Number of Patches:** $N = 256$.
- **Transformer Heads:** $H = 8$.
- **Transformer Layers:** $L = 2$.
- **MLP Hidden Units:** $[256, 128]$.
- **Dropout Rate:** $\gamma = 0.2$.

Training

Several techniques have been introduced to optimize the Transformer model for the model. Nguyen and Salazar [2019] proposed a pre-normalization method to stabilize training by improving gradient flow in the Transformer, a critical advancement in training deep networks . Similarly, Xiong et al. [2020] explored layer normalization's impact within the Transformer architecture, further enhancing model robustness .

Training Methodology

1. Data

Given the limited amount of available data, data augmentation techniques were employed to expand the effective dataset size. The data augmentation process was implemented using the following configuration:

```
train_datagen_aug = ImageDataGenerator(
    validation_split=0.05,  % Same validation ratio
    rescale=1./255,
    rotation_range=30,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)
```

The augmented dataset was mixed with the original, untransformed dataset in a 4:1 ratio to create a new training dataset. This combined dataset was then used for model training to improve generalization and robustness.

2. Hyperparameters

The model was trained using the following hyperparameter settings:

- **Optimizer:** Adam optimizer was utilized for its adaptive learning capabilities.
- **Learning Rate:** A fixed learning rate of 0.0001 was used throughout the training process. Previous studies show that learning rate scheduling is critical for transformer model Nguyen and Salazar [2019]. However, due to the lack of time, I did not try for that.
- **Batch Size:** The batch size was set to 32, which is the maximum value that the available hardware (a laptop) could handle since larger batch size shows better validation accuracy for transformer models. Nguyen and Salazar [2019]
- **Train-Validation Split:** The train-validation split ratio was set to 0.95 : 0.05. This choice was made to maximize the amount of data used for training, given the already limited dataset size. Despite the small validation set, the validation loss and validation accuracy metrics were monitored to ensure reliable performance evaluation.
- **Activation Function:** I utilized GELU(Gaussian Error Linear Unit) introduced by Hendrycks and Gimpel [2023] since it shows low validation loss compared to ReLU.

3. Model Hyper Parameters

Table 1 shows the architecture of the model that achieved the lowest validation loss after multiple experiments. The total number of parameters is 400,000, and Figure 1 illustrates the training and validation loss as well as the accuracy of the model for 90 training epochs. The lowest validation loss occurs on epoch 73.

I conducted various experiments based on the architecture of a baseline model consisting of 4 CNN layers and 2 Transformer layers. The sequence of experiments is as follows:

1. Changing the kernel size of a specific CNN layer from 3×3 to 5×5,
2. Adding 1-2 CNN layers
3. Reducing the number of CNN layers by 1-2 and adding an average pooling layer
4. Increasing the number of channels in a specific CNN layer
5. Reducing the number of channels in a specific CNN layer
6. Adding 1 Transformer layer
7. Reducing the number of Transformer layers to 1
8. Changing the number of attention heads in the Transformer layer to 4
9. Increasing the dimension of the Transformer model
10. Decreasing the dimension of the Transformer model

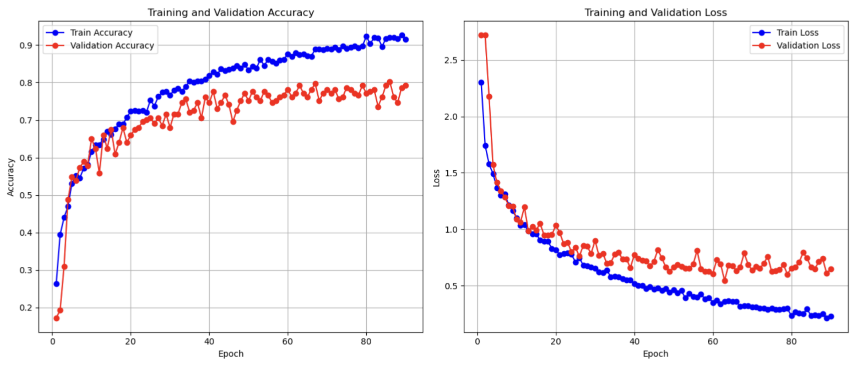

Figure 1: validation, train accuracy and loss

11. Increasing the input data size to a value between 256 and 448 (resolution increase).

The hypotheses for these experiments are as follows:

1. If the current model size is smaller than the capacity required for the dataset, validation loss should decrease when the resolution, model depth, and model width are proportionally increased, assuming the model's depth and width are optimized. Tan and Le [2020]

2. If the current model size is excessively large for the dataset, the risk of overfitting increases, requiring a reduction in model size.

After conducting all 11 experiments, the lowest validation loss at 20 epochs did not show a significant difference or was higher compared to the baseline model. Therefore, the baseline model was selected as the final model.

Table 1: CHAE Baseline Model Architecture

Layer Type	**Output Shape**	**Details**
Input Layer	$256 \times 256 \times 3$	Input image
CNN Layers	$256 \times 256 \times 32$	Batch Normalization → Conv2D (3×3) → GELU Activation → MaxPooling
	$128 \times 128 \times 32$	Batch Normalization → Conv2D (3×3) → GELU Activation → MaxPooling
	$64 \times 64 \times 64$	Batch Normalization → Conv2D (3×3) → GELU Activation → MaxPooling
	$32 \times 32 \times 128$	Batch Normalization → Conv2D (3×3) → GELU Activation → MaxPooling
Reshape Layer	256×128	Reshape from $32 \times 32 \times 128$ to 256×128
Transformer Layers Layer Normalization → MLP → Add	256×128	Multi-Head Attention → Add
Layer Normalization → MLP → Add	256×128	Multi-Head Attention → Add
Output Layer	$1 \times$ num_classes	Dense Layer → Softmax Activation

Conclusion

The **CHAE** model effectively combines convolutional feature extraction with transformer-based attention mechanisms, enabling robust performance even with limited training data. The integration of a CLS token and positional embeddings facilitates the capture of global contextual information, while the lightweight architecture with approximately 400,000 parameters ensures computational efficiency.

References

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021. URL `https://arxiv.org/abs/2010.11929`.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. URL `https://arxiv.org/abs/1512.03385`.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus), 2023. URL `https://arxiv.org/abs/1606.08415`.

Toan Q. Nguyen and Julian Salazar. Transformers without tears: Improving the normalization of self-attention. 2019. doi: 10.5281/ZENODO.3525484. URL `https://zenodo.org/record/3525484`.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015. URL `https://arxiv.org/abs/1409.1556`.

Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks, 2020. URL `https://arxiv.org/abs/1905.11946`.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. URL `https://arxiv.org/abs/1706.03762`.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tie-Yan Liu. On layer normalization in the transformer architecture, 2020. URL `https://arxiv.org/abs/2002.04745`.