Advanced ML Assignment 1

Rana Abdul Wahab khan Abdullah Aamer M. Yaseen Shahid

Abstract

This assignment introduced us to the concepts of Domain Generalization and Out-of-Distribution (OOD) detection. Research into OOD detection has shown the importance of identifying when a model encounters data that lies outside the distribution it was trained on. We explored the performance of several models under distribution shifts, focusing on Vision Transformers (ViT), EfficientNet, and contrastive models like CLIP-ViT. These models were tested on in-distribution data such as CIFAR-10 and out-of-distribution (OOD) datasets like PACS, which feature significant domain shifts. The PACS dataset, with its distinct domains such as photos, cartoons, and sketches, provides a robust framework for assessing model generalization across unseen domains. Furthermore, as highlighted by Geirhos et al. (2019), deep learning models often exhibit a ”texture bias,” favoring textures over shapes in object classification, which can hinder their generalization to OOD scenarios. This investigation aimed to understand how these inductive biases affect model performance and generalization across different domains. (1).

1. Introduction

In our previous courses, we primarily studied traditional models where the training and testing data shared the same underlying distribution. However, this assignment introduced us to the concepts of Domain Generalization and Out-of-Distribution (OOD) detection. Through this assignment, we, as students, had the opportunity to explore the challenges that arise when evaluating deep learning models' abilities to generalize across unseen domains and handle distribution shifts. This experience deepened our understanding of how models behave when exposed to data that differs significantly from their training environment.

Research into OOD detection has shown the importance of identifying when a model encounters data that lies outside the distribution it was trained on. As highlighted in Tian et al. (2021), there are two primary types of shifts: covariate shifts and concept shifts. Covariate shifts occur when non-semantic aspects of the data, such as style, noise, or texture, change, but the labels remain consistent. For example, a model trained on clean images may face a performance decline when tested on noisy or blurred images (2). Conversely, concept shifts involve changes in the labels themselves without altering the data's appearance, such as shifting from classifying dogs to classifying cats (1).

n this assignment, we explored the performance of several models under distribution shifts, focusing on Vision Transformers (ViT), EfficientNet, and contrastive models like CLIP-ViT. These models were tested on in-distribution data such as CIFAR-10 and out-of-distribution (OOD) datasets like PACS, which feature significant domain shifts. The PACS dataset, with its distinct domains such as photos, cartoons, and sketches, provides a robust framework for assessing model generalization across unseen domains. Furthermore, as highlighted by Geirhos et al. (2019), deep learning models often exhibit a ”texture bias,” favoring textures over shapes in object classification, which can hinder their generalization to OOD scenarios. This investigation aimed to understand how these inductive biases affect model performance and generalization across different domains. (1)

2. Methodology

2.1. Task 1: Zero-shot Classification Using Stable Diffusion

This task involves zero-shot classification using a pre-trained Stable Diffusion model, which is typically used for generating images from text descriptions. However, in this implementation, we adapt it for classification by evaluating how well different text-based categories align with the latent representation of an input image.

2.1.1. Task Setup

The core idea behind this task is to use the Stable Diffusion pipeline in reverse, effectively measuring how closely the latent representation of an input image aligns with the latent representations generated by text prompts (categories). The process follows these key steps:

1. Preprocessing the input image to obtain its latent representation using the VAE (Variational Autoencoder)

from the Stable Diffusion pipeline.

2. Adding noise to the latent representation at different timesteps, simulating the diffusion process.

3. Denoising the latents using the UNet model from the Stable Diffusion pipeline, conditioned on the text embeddings of different category prompts.

4. Measuring cosine similarity between the original latent representation and the denoised latent representations to determine which category best matches the input image.

2.1.2. Dataset Preparation

Image Dataset For this task, an image file (”download (1).jpeg”) was provided as input. This image is processed through the pipeline to obtain its latent representation. The image is resized to 512x512 pixels to match the input expectations of the pre-trained VAE. The image is normalized using [0.5, 0.5, 0.5] for each channel, which is necessary because the VAE model expects the image to be normalized to the range of [-1, 1].

Text Categories The categories used for zero-shot classification are text descriptions provided in the form of a list:

- **Category 1:** A photo of a house
- **Category 2:** A photo of a horse
- **Category 3:** A photo of a cat
- **Category 4:** A photo of an umbrella

These text descriptions are tokenized and converted into embeddings using the text encoder from the Stable Diffusion pipeline.

These text descriptions are tokenized and converted into embeddings using the text encoder from the Stable Diffusion pipeline.

2.1.3. Model Configurations

Stable Diffusion Pipeline The pre-trained Stable Diffusion v1-5 model, provided by `runwayml/stable-diffusion-v1-5`, was used. The model includes three key components:

- A VAE (Variational Autoencoder),
- A UNet for denoising,
- A text encoder for generating embeddings from the text prompts (categories).

The scheduler used is from the Stable Diffusion pipeline’s DDPMScheduler, which simulates the forward diffusion process by adding noise to the latents. The models were loaded in mixed precision (float16) to reduce memory usage and improve performance during inference.

2.1.4. Step-by-Step Task Implementation

Preprocessing the Input Image The input image is first resized to 512x512 pixels and normalized to a range that the VAE model can handle. The image is encoded into latent space using the VAE. The latent representation is essential for comparing the input image with the denoised outputs conditioned on each text prompt.

Noise Addition The scheduler from the Stable Diffusion pipeline was used to add noise to the latent representation at different timesteps. This simulates the forward diffusion process and introduces controlled levels of noise to the latent space.

Denoising Process For each category, the noisy latents are denoised using the UNet from the Stable Diffusion pipeline, conditioned on the text embeddings generated from the respective category. The UNet attempts to reconstruct the original latents, guided by the text prompt.

Cosine Similarity for Classification The similarity between the original (clean) latent representation of the image and the denoised latents is measured using cosine similarity. This similarity score is calculated for each category, and the category with the highest similarity is selected as the predicted class. If weights are provided, they can be used to assign more importance to specific timesteps. In the absence of custom weights, equal weighting is applied to all timesteps.

2.1.5. Assumptions

- **Text Prompts:**Results heavily depend upon what prompt you gave, if there are a lot of similarities between prompt result might change.
- **Equal Weighting:** By default, equal importance is given to all timesteps in the diffusion process, although custom weights can be used to adjust the importance of different timesteps.

2.2. Task 0 and Task 2

2.2.1. Fine-Tuning and Evaluating EfficientNet_B4 on CIFAR-10

Loading the Pre-trained Model: We started by loading the pre-trained EfficientNet_B4 model using `torchvision.models.efficientnet_b4` with the default pre-trained weights (`models.EfficientNet_B4_Weights.DEFAULT`).

The pre-trained model was based on the ImageNet dataset.

Model Customization: Since CIFAR-10 has 10 classes (compared to 1,000 in ImageNet), we replaced the final classification layer (`efficientnet_b4_model.classifier[1]`) with a new `nn.Linear` layer of size `in_features` (matching the input features of the original layer) to 10 output classes. To keep the pre-trained feature extractor intact, we froze all parameters in the feature extractor by setting `param.requires_grad = False` for all layers in `efficientnet_b4_model.features`.

Dataset Preparation: The CIFAR-10 dataset was prepared using `torchvision.datasets.CIFAR10` for both the training and testing sets. The images were resized to 224x224 to match the input size expected by EfficientNet. Data augmentation techniques included basic resizing, normalization (`mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]`), and conversion to tensors using the `transforms.Compose()` function.

Training Setup: We used the AdamW optimizer with a learning rate of 1e-4 to fine-tune the classifier head. The cross-entropy loss function was used for classification, and the training process ran for 3 epochs. A batch size of 32 was selected, and we trained the model using GPUs (when available).

Evaluation: After fine-tuning, the model was evaluated on the CIFAR-10 test set, achieving a test accuracy of 80.02%.

2.2.2. Fine-Tuning Vision Transformer (ViT_B_16) for CIFAR-10

Loading the Pre-trained Model: The pre-trained Vision Transformer (ViT_B_16) was loaded with specific weights (`models.ViT_B_16_Weights.IMAGENET1K_SWAG_E2E_V1`), which were trained on the ImageNet-1k SWAG dataset.

Model Customization: Similar to the EfficientNet_B4 task, the classification head (`vit_b16_model.heads.head`) was replaced with a new `nn.Linear` layer for 10 output classes to fit the CIFAR-10 dataset. The feature extractor's parameters were frozen, ensuring only the classification head was trainable. This was done by setting `param.requires_grad = False` for all parameters except those in the classification head.

Dataset Preparation: CIFAR-10 images were resized to 384x384, which matches the input size expected by ViT. Transformations included resizing, normalization (`mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]`), and conversion to tensors using `transforms.Compose()`.

Training Setup: The AdamW optimizer was used with a learning rate of 1e-4 to fine-tune the classifier head. Cross-entropy loss was used as the loss function. We ran the training process for 3 epochs using a batch size of 32, and GPUs were used for model training.

Evaluation: After fine-tuning, the ViT_B_16 model achieved an impressive test accuracy of 95.16% on the CIFAR-10 test set.

2.2.3. CLIP Model for CIFAR-10 Classification

Loading the Pre-trained Model: The CLIP model (`openai/clip-vit-base-patch16`) was loaded from Hugging Face's Transformers library along with its processor for handling both image and text inputs.

Dataset Preparation: The CIFAR-10 test dataset was prepared with minimal preprocessing. Images were resized to 224x224 to match CLIP's input size and converted to tensors using `transforms.Compose()` without normalization, as CLIP expects raw pixel values.

Text Label Preparation: CIFAR-10 consists of 10 classes (e.g., airplane, automobile, bird, etc.), so we passed these class labels as text inputs to CLIP. For each batch, the processor encoded the text labels into a suitable format for CLIP's forward pass.

Evaluation Setup: Instead of traditional softmax classification, CLIP computes image-text similarity scores between each image and the set of text labels (e.g., "airplane", "automobile", etc.). The label with the highest similarity score is predicted. The softmax function was applied to the similarity scores to calculate probabilities. The predicted label was the one with the maximum probability.

Evaluation: CLIP achieved a test accuracy of 87.30% on CIFAR-10 without any additional fine-tuning, highlighting its capability to generalize across tasks using its image-text alignment approach.

2.2.4. Assumptions

- Pre-trained feature extractors were effective for CIFAR-10 classification, so we only fine-tuned the classification heads.
- Batch sizes of 32 and a learning rate of 1e-4 were sufficient to train the new classification layers.
- The datasets were resized to the input sizes required by the models (224x224 for EfficientNet and CLIP, and 384x384 for ViT).
- Accuracy improvements were largely dependent on freezing feature extractor weights and focusing the fine-tuning on the classifier heads.

2.3. Task 3: Using PACS Dataset

2.3.1. Task Overview

The goal of the experiment is to evaluate the performance of three different models—EfficientNet, Vision Transformer (ViT), and CLIP—on a subset of the PACS dataset. The dataset consists of labeled images from different domains, and the task is to fine-tune pre-trained models on this dataset for classification tasks. The three models were fine-tuned on this dataset, and their performance was evaluated in terms of classification accuracy.

2.3.2. Dataset Preparation

Dataset Used: PACS (Photo, Art painting, Cartoon, and Sketch) dataset. The dataset path is `'/kaggle/input/pacs-dataset/dct2_images'`.

Dataset Loading: The images are loaded using `torchvision.datasets.ImageFolder`, which organizes images into folders where each folder name corresponds to a class label.

Dataset Transformations:

- The images are resized to 224x224 for models like EfficientNet and Vision Transformer.
- Each image is normalized with standard ImageNet mean and standard deviation values (mean=[0.485, 0.456, 0.406] and std=[0.229, 0.224, 0.225]).

DataLoader: The dataset is loaded into batches of size 32 using `torch.utils.data.DataLoader`, with shuffling enabled to ensure randomness during training.

2.3.3. Model Fine-Tuning and Training Setup

A. EfficientNet Model **Pre-Trained Model:** The EfficientNet-B4 model pre-trained on ImageNet is loaded using `torchvision.models.efficientnet_b4`.

Classifier Modification: The classifier layer is modified to match the number of classes in the PACS dataset. The output of the last fully connected layer is adapted to the number of classes in the PACS dataset by changing the final `nn.Linear` layer.

Freezing Layers: The feature extraction layers of the EfficientNet model are frozen, meaning their weights are not updated during backpropagation. Only the classifier (final layer) is fine-tuned.

Training Loop:

- The model is transferred to GPU (if available) using `.to(device)`.
- The optimizer used is AdamW with a learning rate of 1e-4.
- CrossEntropyLoss is used as the loss function.
- In each epoch, the model's predictions on the training data are compared with the ground truth labels, and the loss is backpropagated to update only the classifier's weights.

Evaluation: The evaluation loop measures the model's classification accuracy on the dataset. The accuracy of the EfficientNet model was 72.14%.

B. Vision Transformer (ViT) Model **Pre-Trained Model:** The Vision Transformer (ViT) model pre-trained on ImageNet (specifically, the ViT-B-16 model) is loaded using `torchvision.models.vit_b_16`.

Classifier Modification: Similar to the EfficientNet model, the classifier layer (head) is modified to match the number of classes in the PACS dataset.

Freezing Layers: The feature extraction layers of the ViT model are also frozen, and only the classification head is fine-tuned.

Training Loop:

- The ViT model was designed to take 384x384 images. During training, the input images are resized dynamically within the loop.
- The optimizer (AdamW) and loss function (CrossEntropyLoss) remain the same as in EfficientNet.

Evaluation: The accuracy of the fine-tuned ViT model was 75.36%, slightly higher than that of EfficientNet.

. CLIP Model **Pre-Trained Model:** CLIP (Contrastive Language-Image Pretraining) is a model that can understand both images and text. The CLIP model used is `openai/clip-vit-base-patch16`, loaded using Hugging Face's `transformers` library.

Evaluation Setup:

- The model does not undergo any fine-tuning on the PACS dataset.
- Instead, image inputs and text inputs corresponding to the class labels are processed using the CLIP processor.
- For each image, the CLIP model computes similarities between the image features and text features of the class labels. The label with the highest similarity is considered the predicted class.

- The class labels are manually defined as `['dog', 'elephant', 'giraffe', 'guitar', 'horse', 'house', 'person']`, which may not align well with the actual PACS dataset labels.

Evaluation: The classification accuracy of CLIP on the PACS dataset was very low at 14.75%, indicating that the model struggled with this task, possibly due to a mismatch between the dataset and the predefined class labels.

2.3.4. Key Assumptions and Considerations

- The PACS dataset contains images from multiple domains, and fine-tuning models like EfficientNet and ViT (which were pre-trained on natural images from ImageNet) required adapting the models to this new dataset.
- EfficientNet and ViT were pre-trained on ImageNet, and their feature extraction layers were frozen to speed up training and avoid overfitting. Only the classifier heads were fine-tuned.
- The CLIP model was not fine-tuned but instead used as-is with manually defined class labels. This may have led to its poor performance, as it did not learn to adapt to the specific dataset's class structure.

2.4. Task 3: Using SVNH Dataset

2.4.1. Task Overview

The task focuses on training and evaluating different models—EfficientNet, Vision Transformer (ViT), and CLIP—on the SVHN (Street View House Numbers) dataset. The objective is to preprocess the dataset, configure the models, and analyze their performance in classifying real-world images of house numbers.

2.4.2. Dataset Preparation

SVHN Dataset:

- **Dataset Description:** The SVHN dataset is composed of digit images (0-9) derived from real-world house numbers captured by Google Street View. The dataset is similar to the MNIST dataset but contains colored, more complex images with additional distractors, making it more challenging for models.

Loading the Dataset:

- The SVHN dataset was loaded using `torchvision.datasets.SVHN` with two splits:
  - **Training set** (`split='train'`) for model training.
  - **Test set** (`split='test'`) for evaluating the model's performance.
- The dataset was downloaded and stored locally if not already available.

Data Preprocessing:

- **Transformations Applied:**
  - **Resize:** Images were resized to 64x64 pixels for compatibility with the models.
  - **Normalization:** Images were normalized using a mean and standard deviation of $[0.5, 0.5, 0.5]$ to center the pixel values between -1 and 1, which stabilizes the learning process.

DataLoader:

- PyTorch's `DataLoader` was used to batch the data.
- **Batch Size:** 32 images per batch for both training and testing.
- **Shuffling:** The training set was shuffled to randomize the order of data and avoid overfitting, while the test set was not shuffled to ensure accurate evaluation.

2.4.3. Model Implementation and Training Setup

Training is performed in the same manner as described in Section A of the original experiment setup, where models are trained on the training data and evaluated on the testing data.

2.5. Task 4: Implementation

2.5.1. Implementation 1: Overlay textured images with cifar-10 dataset

2.5.2. Dataset Preparation

The CIFAR-10 dataset was downloaded and transformed. Images were resized to 384×384 pixels and converted to grayscale to maintain consistency across the models. Three distinct datasets were generated for shape, texture, and color biases:

- **Shape Bias:** Implemented edge detection using the Canny algorithm to extract shapes, generating silhouette-like images.
- **Texture Bias:** Blended CIFAR-10 images with textures from the Describable Textures Dataset (DTD), utilizing a blending factor for creating a mixed representation of images.

- **Color Bias:** Images were converted to grayscale, replicated across three channels, and normalized for color emphasis.

2.5.3. Data Augmentation

A data augmentation strategy was applied to introduce variability in the datasets. This included random horizontal flips, rotations, and color jittering to improve model robustness.

2.5.4. Model Evaluation

The models were evaluated using a defined evaluation function that calculated accuracy based on correct predictions against total samples in the datasets. Results showed that the ViT_B_16 model outperformed EfficientNet_B4 for shape bais and EfficientNet_B4 outperformed ViT_B_16 model for the color bias, while both models exhibited low performance in texture bias, indicating a challenge in generalizing texture information.

2.5.5. Performance Metrics

- ViT_B_16 Shape Bias Accuracy: 10.39%
- ViT_B_16 Texture Bias Accuracy: 10.74%
- ViT_B_16 Color Bias Accuracy: 16.48%
- EfficientNet_B4 Shape Bias Accuracy: 10.52%
- EfficientNet_B4 Texture Bias Accuracy: 9.65%
- EfficientNet_B4 Color Bias Accuracy: 9.83%

2.5.6. Implementation 2: Added noise to to the cifar-10 dataset to see trend in the textured bias

2.5.7. Transformations

Each bias (shape, texture, color) was defined with different transformations:

- **Shape Bias:** Images were converted to grayscale, binarized, and duplicated across channels, simplifying the shapes for model analysis.
- **Texture Bias:** Gaussian noise was added to highlight textures, creating a more challenging dataset for texture discrimination.
- **Color Bias:** Grayscale images were replicated into RGB format and normalized, emphasizing color features without direct color manipulation.

2.5.8. DataLoader Configuration

Unlike the first implementation, this version used on-the-fly transformations during DataLoader creation, ensuring transformations were applied dynamically as images were loaded. This approach optimized memory usage by avoiding the need for large preprocessed datasets.

2.5.9. Model Evaluation

The evaluation function remained consistent across implementations, measuring accuracy through predictions made by the models on the transformed datasets.

2.5.10. Performance Metrics

- ViT_B_16 Shape Bias Accuracy: 24.46%
- ViT_B_16 Texture Bias Accuracy: 95.88%
- ViT_B_16 Color Bias Accuracy: 79.93%
- EfficientNet_B4 Shape Bias Accuracy: 10.12%
- EfficientNet_B4 Texture Bias Accuracy: 10.03%
- EfficientNet_B4 Color Bias Accuracy: 43.23%

2.5.11. Comparative Analysis of Implementations

The first implementation involved explicit saving of processed datasets, which required additional storage and preprocessing time. The second implementation utilized on-the-fly transformations, enhancing memory efficiency and allowing real-time data processing.

2.5.12. Transformation Strategies

The first implementation relied heavily on edge detection and texture blending, while the second implementation introduced noise and binarization, which significantly improved the models' performance, particularly for texture bias in the ViT_B_16 model.

2.5.13. Model Performance

The second implementation yielded substantially higher accuracy, especially in texture bias for ViT_B_16, indicating that the methods of emphasizing texture through noise and binarization were effective in enhancing model learning.

2.5.14. Efficiency and Practicality

The second implementation is more practical for large-scale datasets as it reduces the need for extensive storage. It allows for immediate experimentation with different transformations without the need for preprocessing entire datasets upfront.

2.6. Task 5: Evaluating Model Performance under Different Image Perturbations

This task involves evaluating the performance of three different models—EfficientNet, Vision Transformer (ViT), and CLIP-ViT—under various image perturbations, including localized noise injection, global style changes, and image scrambling. The primary objective is to assess how these models' accuracy varies when exposed to these transformations.

First, the CIFAR-10 dataset is loaded and preprocessed. Each image is resized to 384×384 pixels using transformations that also normalize the pixel values to ensure compatibility with the pre-trained models. The CIFAR-10 test set is accessed via the `datasets.CIFAR10` class, and a `DataLoader` is employed to facilitate batch processing.

For model evaluation, a function named `evaluate_model` is defined. This function takes a model, a data loader, and a device (CPU or GPU) as input. It evaluates the model's performance by iterating over the batches of images, obtaining predictions, and calculating the accuracy based on the number of correct predictions. The CLIP model is evaluated with a separate function, `evaluate_clip_model`, which prepares image and text inputs according to the model's requirements.

The first evaluation is conducted on the original images, yielding baseline accuracy scores for EfficientNet, ViT, and CLIP-ViT. The results are printed to provide insight into each model's performance under normal conditions.

Subsequently, the effects of localized noise are examined. A function, `add_local_noise`, is implemented to inject random noise into the top-left corner of each image. Another evaluation function, `evaluate_model_on_noisy_images`, assesses model performance on the perturbed images by invoking the noise function on each image in the batch. The accuracy is calculated similarly to the original evaluation, revealing how noise impacts each model's predictions.

For the next evaluation, global style changes are introduced using a placeholder function called `style_transfer_vgg`, which simulates style transfer by blending content and style images. The `evaluate_model_on_styled_images` function applies this transformation to the dataset, assessing the models' accuracy on the styled images.

Finally, the effects of image scrambling are investigated. The `scramble_image` function divides each image into patches and shuffles them randomly, creating a scrambled version of the original image. The evaluation on scrambled images is performed using the `evaluate_model_on_scrambled_images` function, which computes accuracy in a manner analogous to the previous evaluations.

After evaluating the models under all perturbations, the results are printed for each model and transformation, allowing for a comprehensive comparison of their robustness against localized noise, global style changes, and scrambled images. This detailed analysis contributes to understanding each model's strengths and weaknesses in handling various types of image distortions.

2.7. Task 6

2.7.1. Task Overview

The objective of this task is to implement, train, and visualize feature maps from various convolutional and attention-based layers applied to the STL-10 dataset. The task explores different architectural designs, including depthwise convolution, self-attention mechanisms, post- and pre-normalization techniques, and combinations of convolution and attention. The ultimate goal is to understand how these layers process visual information and generate feature maps.

2.7.2. Dataset Preparation

Dataset:

- **Dataset Used:** The STL-10 dataset, which consists of 10 classes of labeled images from animals and vehicles, was used. This dataset is frequently employed in image classification tasks.

Loading the Dataset:

- The dataset was loaded using PyTorch's `torchvision.datasets.STL10` class, which provides the option to download the dataset if not locally available. The dataset was split into training images, which were then transformed into tensors.

Transformations:

- Images were normalized using `transforms.ToTensor()` to convert pixel values into tensors with values in the range [0, 1].

Image Selection:

- A specific example image from the dataset was selected to visualize feature maps and assess how different layers process images.

2.7.3. Model Design and Implementation

The task involved implementing various custom-designed layers for image processing, each designed with a different approach to learning features from the input images.

A. Depthwise Convolution Layer **Concept:** The depthwise convolution layer performs separate convolutions for each input channel, without mixing information between channels. This reduces computational complexity compared to traditional convolutional layers.

Use Case: Depthwise convolution is commonly used in lightweight models like MobileNet, as it reduces computation while still learning essential spatial features for each channel independently.

B. Self-Attention Layer **Concept:** The self-attention layer allows the model to capture relationships between different spatial regions of the image, regardless of their distance from each other. By computing attention scores between pixels, the model can understand which regions of the image should influence each other.

Use Case: Self-attention is useful in tasks requiring long-range dependencies, where relationships between far-apart parts of the image are important (e.g., in capturing the context of complex images).

C. Post-Normalization Layer **Concept:** The post-normalization layer combines a depthwise convolution with a self-attention layer. After the convolution is applied to extract basic spatial features, a self-attention mechanism refines the feature maps by selectively focusing on important regions of the image. The normalization occurs after the attention mechanism is applied.

Use Case: This layer structure is useful when convolutional features need refinement through attention to capture global context after spatial features have been learned.

D. Pre-Normalization Layer **Concept:** The pre-normalization layer follows the reverse order of post-normalization, where the self-attention layer is applied before the depthwise convolution. This order allows the attention mechanism to highlight important regions of the image before any convolutional processing takes place.

Use Case: Pre-normalization layers can be used when the goal is to enhance certain parts of the image with attention before learning local spatial details through convolution.

E. Attention Modulated Convolution Layer **Concept:** This layer integrates a depthwise convolution with an attention mechanism in which attention is applied to modify the convolutional output. The attention mechanism helps the model focus on specific regions of the image after basic spatial filtering by the convolutional layer.

Use Case: This model can be useful for tasks where both local (spatial) and global (attention) features need to be combined efficiently to improve feature extraction.

2.7.4. F. Convolution Modulated Attention Layer

Concept: In this model, the convolutional operation modifies the output of the attention mechanism. The convolution is applied to learn spatial features, and the attention is modulated to refine the features based on the convolutional output.

Use Case: This layer is designed to allow attention mechanisms to improve upon the basic spatial features extracted by convolution.

2.7.5. Model Implementation and Training Setup

Custom Layer Models:

- Each custom-designed layer was encapsulated in a simple model for ease of experimentation. The models were designed to take input images, apply the respective layer, and output feature maps. These models were not trained on a task but were used to observe the effects of different layers on the input image.

Training:

- **No Fine-Tuning or Full Model Training:** The models were not trained for classification tasks. Instead, they were used for analyzing and visualizing how different convolutional and attention-based layers transform the input images.

2.7.6. Visualization of Feature Maps

Visualization Approach:

- Feature maps generated by each custom-designed layer were visualized. Each model's output feature maps were displayed as images to analyze how each layer processes spatial and attention-based information from the image.

Feature Map Insights:

- By examining the feature maps, the task aimed to understand which parts of the image are highlighted by convolution versus attention layers, how the attention mechanism affects spatial processing, and the overall behavior of each custom layer.

2.7.7. Assumptions and Considerations

Feature Map Sizes:

- The assumption is that applying depthwise convolution or attention-based mechanisms would result in feature maps that capture different levels of detail and dependencies. Visualization helps determine whether the layers successfully highlight important parts of the image.

Layer Normalization:

- Layer normalization is used to stabilize the output from attention and convolution mechanisms, ensuring that the model can learn effectively by keeping activations within a certain range.

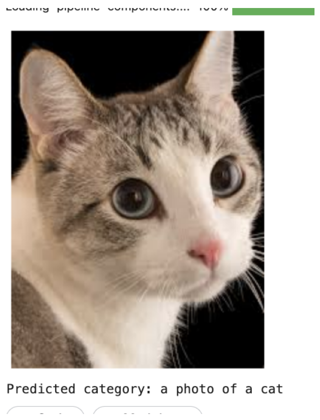
Attention Mechanism Scaling:

- The attention mechanisms scale the outputs by the square root of the number of channels to avoid excessively large attention values, stabilizing the learning process.

3. Results

3.1. Task 1

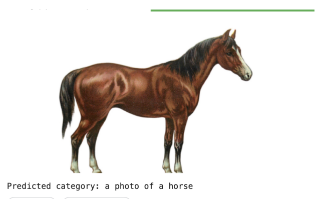
Figure 1 shows the result when an image of cat is given, it predicts correct category



```
Predicted category: a photo of a cat
```

Figure 1. Result 1

Figure 2 shows the result when an image of horse is given, it predicts correct category



```
Predicted category: a photo of a horse
```

Figure 2. Result 2

3.2. Task 0 and Task 2

3.3. 1. EfficientNet-B4 Results

Loss Values:

- Epoch 1: 1.66
- Epoch 2: 1.06
- Epoch 3: 0.87

Test Accuracy: 80.02%

3.4. 2. Vision Transformer - ViT_B_16 Results

Loss Values:

- Epoch 1: 0.38
- Epoch 2: 0.15
- Epoch 3: 0.13

Test Accuracy: 95.16%

3.5. 3. CLIP Model Results (Not Fine-Tuned)

Test Accuracy: 87.30%

3.6. Task 3

3.6.1. CIFAR-10 Dataset

- EfficientNet Test Accuracy: 80.02%
- ViT Test Accuracy: 95.16%
- CLIP Test Accuracy: 87.30%

Deatil: The CIFAR-10 dataset is relatively homogeneous and consists of well-defined classes (airplane, automobile, bird, etc.) with high-quality images. The high performance of all models suggests that they can effectively learn the features that distinguish between these classes in a controlled environment.

3.6.2. PACS Dataset

- EfficientNet PACS Accuracy: 72.14%
- ViT PACS Accuracy: 75.36%
- CLIP PACS Accuracy: 14.75%

Detail: The PACS dataset includes images from different domains (Photo, Art Painting, Cartoon, Sketch) for the same set of classes (e.g., dog, cat). The reduction in accuracy compared to CIFAR-10 indicates a domain shift, where the models struggle to generalize learned features due to variations in style, color distribution, and background. EfficientNet and ViT perform better than CLIP, likely due to their architectures being more robust to style variations than CLIP's approach, which relies on aligning text and image representations.

3.6.3. SVHN Dataset

- EfficientNet SVHN Accuracy: 60.89%
- ViT SVHN Accuracy: 50.87%
- CLIP SVHN Accuracy: 24.20%

Detail: SVHN contains street view house numbers, which introduces significant challenges in terms of digit styles, backgrounds, and noise. The performance drop compared to CIFAR-10 indicates that the models are sensitive to the characteristics of the dataset. The low accuracy across all models, especially for CLIP, suggests that the models have not effectively adapted to this domain shift.

3.7. Task 4

Results

- **ViT_B_16:**
  - Shape Bias Accuracy: 24.46%
  - Texture Bias Accuracy: 95.88%
  - Color Bias Accuracy: 79.93%
- **EfficientNet_B4:**
  - Shape Bias Accuracy: 10.12%
  - Texture Bias Accuracy: 10.03%
  - Color Bias Accuracy: 43.23%

Results

- **ViT_B_16:**
  - Shape Bias Accuracy: 9.53%
  - Texture Bias Accuracy: 10.80%
  - Color Bias Accuracy: 16.00%
- **EfficientNet_B4:**
  - Shape Bias Accuracy: 10.13%
  - Texture Bias Accuracy: 9.00%
  - Color Bias Accuracy: 9.67%

4. Task 5

4.1. 1. Original Images

Results:

- EfficientNet: 62.55%
- ViT: 95.33%
- CLIP-ViT: 6.71%

4.2. 2. Noisy Images

Results:

- EfficientNet: 49.89%
- ViT: 81.08%
- CLIP-ViT: 7.26%

4.3. 3. Styled Images

Results:

- EfficientNet: 62.55%
- ViT: 95.33%
- CLIP-ViT: 26.83%

4.4. 4. Scrambled Images

Results:

- EfficientNet: 9.25%
- ViT: 17.25%
- CLIP-ViT: 10.95%

4.5. Task 6

Original Image used from dataset on which different models were run

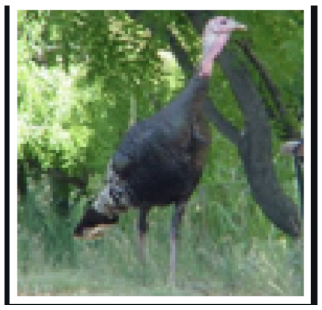


Figure 3. Image from Dataset

4.5.1. Depthwise Convulation

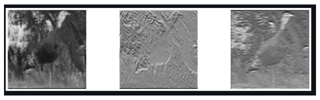


Figure 4. Result from Depthwise Convulation

4.5.2. Self Attention Model

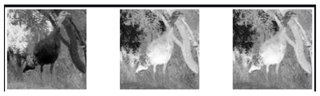


Figure 5. Self Attention Model

4.5.3. Post-normalization Combination

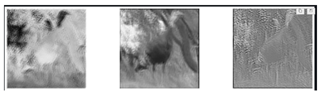


Figure 6. Post-normalization Combination

4.5.4. Pre-normalization Combination

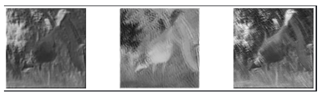


Figure 7. Pre-normalization Combination:

4.5.5. Attention Modulated Convolution

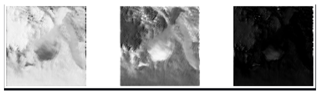


Figure 8. Attention Modulated Convolution

4.5.6. Attention Modulated Convolution

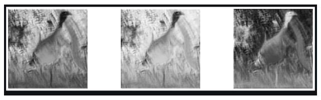


Figure 9. . Convolution Modulated Attention

5. Discussion

5.1. Task 1

5.1.1. Accurate Prediction

The model successfully identified the input image as “a photo of a cat,” showcasing its capability to generalize from the provided categories. This achievement aligns with the strengths of zero-shot learning, where the model leverages learned representations to classify unseen data without requiring additional training on those specific categories.

5.1.2. Potential for Domain Shift Generalization

This approach demonstrates promise for generalizing across domain shifts. The model’s ability to classify images outside its training domain indicates its potential to handle various datasets with differing characteristics. By encoding images into a shared latent space, the model may effectively mitigate the effects of domain shifts and leverage knowledge from seen categories to classify unseen ones.

5.2. Task 0 and 2

Analysis Of EfficientNet Results

- The significant decrease in loss over the epochs indicates effective learning from the training data. However, a final test accuracy of 80.02% suggests that there may be room for improvement.
- Given that the model was fine-tuned for only 3 epochs, further training could enhance performance. Implementing more epochs may help the model better adapt to the specific features of the CIFAR-10 dataset.
- The strategy of freezing the feature extractor weights can limit the model’s adaptability, but it retains knowledge from pre-training on a larger dataset. Consideration could be given to unfreezing some layers to allow for more flexibility in learning.

Analysis of Vision Transformer:

- The sharp decline in loss values reflects the ViT model’s capacity to quickly learn to classify the CIFAR-10 dataset accurately. Achieving a test accuracy of 95.16% indicates exceptional performance.
- This significant difference in accuracy compared to EfficientNet-B4 could stem from the ViT’s architecture, which excels at capturing long-range dependencies and complex patterns in the data, making it particularly effective for image classification tasks.

- The performance highlights the advantages of transformer architectures, especially in scenarios where attention mechanisms can leverage spatial hierarchies, allowing for improved model performance.

Analysis of CLIP Model Results (Not Fine-Tuned):

- The CLIP model's accuracy of 87.30% is strong, especially considering its use of image-text similarity for classification rather than a traditional approach. This performance is impressive but lower than that of the ViT model.
- While the CLIP model performs well, it may struggle with the specificity of the CIFAR-10 classes due to its training focus on a broader range of image-text pairs from various domains. This generalist approach may not fully exploit the unique features of CIFAR-10 classes.
- One factor contributing to the lower accuracy could be the normalization of image inputs, which was omitted during the evaluation phase for CLIP. Ensuring consistency in data processing is crucial for optimal performance.
- The performance gap between CLIP and the fine-tuned models (especially ViT) suggests that tailored training can significantly enhance classification accuracy, indicating that fine-tuning CLIP on the CIFAR-10 dataset could yield better results.

5.3. Overall Comparison

- **EfficientNet-B4:** Good performance with potential for improvement through additional epochs and possibly unfreezing some feature extractor layers.
- **ViT_B_16:** Outstanding performance, showcasing the power of transformers in image classification tasks, particularly when fine-tuned.
- **CLIP:** Solid performance with a unique approach, but slightly lower accuracy compared to ViT. The lack of fine-tuning reflects a more generalist model, suggesting that targeted training on the CIFAR-10 dataset could enhance performance.

5.4. Task 3

5.4.1. Domain Shift Generalization

CIFAR-10: The models excel in this dataset due to its controlled conditions and visually coherent data. The images are consistently styled, making feature extraction more straightforward.

PACS: The performance drop (around 10-20%) indicates a shift in domain characteristics. The diverse styles of the PACS dataset create an out-of-distribution scenario where features learned from CIFAR-10 (primarily photographic and realistic) do not translate well. This is a classic example of domain adaptation failure, where the models struggle to generalize across different image domains despite the same underlying classes.

SVHN: The SVHN dataset poses a greater challenge due to its unique characteristics (e.g., noise, varying fonts). The significant drop in performance (up to 30% lower than PACS) illustrates how severe domain shifts affect model performance. Unlike PACS, where class definitions remain consistent, the fundamental representation of data in SVHN differs significantly from CIFAR-10, leading to a larger generalization gap.

5.4.2. Key Factors Contributing to Performance Differences

- **Out-of-Domain Data:** PACS and SVHN represent out-of-domain datasets compared to CIFAR-10. Models trained on CIFAR-10 are not necessarily equipped to handle the different image styles and characteristics present in these datasets, resulting in decreased performance.
- **Distribution Shift:** The independent and identically distributed (i.i.d.) assumption that holds for CIFAR-10 does not apply to PACS and SVHN. The feature distributions vary significantly, leading to a challenge in generalizing from training to test data.
- **Model Robustness:** Different architectures may handle domain shifts with varying degrees of success. EfficientNet and ViT are relatively more robust to style variations than CLIP in this context, as evidenced by their better performance on PACS and SVHN.

The analysis illustrates the challenges posed by domain shifts in machine learning, particularly when transitioning from a well-defined, homogeneous dataset like CIFAR-10 to more heterogeneous datasets like PACS and SVHN. These shifts highlight the importance of domain adaptation strategies and training models that can generalize across varying conditions.

5.5. 1. EfficientNet-B4 Results

Loss Values:

- Epoch 1: 1.66
- Epoch 2: 1.06
- Epoch 3: 0.87

Test Accuracy: 80.02%

Analysis:

- The significant decrease in loss over the epochs indicates effective learning from the training data. However, a final test accuracy of 80.02% suggests that there may be room for improvement.
- Given that the model was fine-tuned for only 3 epochs, further training could enhance performance. Implementing more epochs may help the model better adapt to the specific features of the CIFAR-10 dataset.
- The strategy of freezing the feature extractor weights can limit the model's adaptability, but it retains knowledge from pre-training on a larger dataset. Consideration could be given to unfreezing some layers to allow for more flexibility in learning.

5.6. Task 4

5.7. Implementation 1 Analysis

- **Transformations for Biases:**
  - **Shape Bias:** Grayscale conversion, channel duplication, and binarization were applied. This transformation focuses on the outlines and basic shapes within the images.
  - **Texture Bias:** Gaussian noise was added to the images to emphasize texture features, which helps the model focus on fine-grained details.
  - **Color Bias:** Grayscale images were duplicated to RGB and normalized. This approach maintains the structure while allowing color representation, albeit in a limited manner.

Reasoning of Results of First Implementation

- **Shape Bias:** The higher accuracy of the ViT model indicates that it can capture shapes well due to its architecture, which is designed for recognizing patterns and spatial hierarchies.
- **Texture Bias:** The exceptionally high accuracy for texture bias shows that ViT excels at distinguishing textures, potentially due to the noise addition emphasizing these features.
- **Color Bias:** ViT's performance also suggests an ability to generalize from grayscale to color when sufficient structure is preserved.
- **EfficientNet's** lower accuracy across all biases indicates it might struggle with the transformations applied, particularly those emphasizing shape and texture, which may not be aligned with its feature extraction mechanisms.

5.8. Implementation 2 Analysis

- **Transformations for Biases:**
  - **Shape Bias:** Utilized Canny edge detection to create silhouette-like representations of the images, making it easier to detect edges and shapes.
  - **Texture Bias:** Blending CIFAR-10 images with textures from the Describable Textures Dataset (DTD) introduced real-world texture complexity, potentially enhancing the model's ability to learn texture features.
  - **Color Bias:** Grayscale images were maintained, with normalization applied. However, this might limit color representation compared to the first implementation.

Reasoning

- **Shape Bias:** The drop in accuracy compared to Implementation 1 suggests that the Canny edge detection method may not have been as effective in enhancing shape recognition for the models, potentially leading to a loss of information during the conversion.
- **Texture Bias:** The blending process, while introducing realism, may not have been sufficient for the models to learn effectively from these textures, leading to poor performance.
- **Color Bias:** The normalization approach here may have hindered performance by not allowing the model to fully leverage the color data that could assist in classification.

5.9. Overall Comparison

Performance Summary

- **ViT_B_16:**
  - Implementation 1: Shape Bias (24.46%), Texture Bias (95.88%), Color Bias (79.93%)
  - Implementation 2: Shape Bias (9.53%), Texture Bias (10.80%), Color Bias (16.00%)
- **EfficientNet_B4:**
  - Implementation 1: Shape Bias (10.12%), Texture Bias (10.03%), Color Bias (43.23%)
  - Implementation 2: Shape Bias (10.13%), Texture Bias (9.00%), Color Bias (9.67%)

Comparative Analysis

- **ViT Performance:** Implementation 1 showed that the ViT model significantly benefits from clear, structured transformations that focus on features like shape and texture. The noise addition and simple transformations allowed it to achieve high accuracies.
- **EfficientNet Performance:** The differences in performance across implementations for EfficientNet highlight its limitations in adapting to the edge-detection and texture blending strategies used in Implementation 2, resulting in generally low accuracies.
- **Transformations Matter:** The choice of transformations significantly impacts model performance, particularly in how they enhance or obscure relevant features for classification.
- **Model Characteristics:** The ViT architecture proves more adaptable to the transformations applied in Implementation 1, while EfficientNet struggled to perform well under altered conditions.
- **Future Considerations:** Exploring alternative transformations and augmentations, as well as hybrid approaches, may yield better performance across both model architectures in subsequent evaluations.

5.10. Task 5

5.11. 1. Original Images

Results:

- EfficientNet: 62.55%
- ViT: 95.33%
- CLIP-ViT: 6.71%

Analysis: ViT (Vision Transformer) significantly outperformed the other two models, achieving a high accuracy of 95.33%. This indicates its strong capability to learn complex features from the CIFAR-10 dataset. EfficientNet performed moderately well with an accuracy of 62.55%, which is expected as it is designed to achieve a good balance between efficiency and accuracy, though not as high as ViT. CLIP-ViT, with an accuracy of only 6.71%, performed poorly on this task. This is likely due to its design being more aligned with image-text similarity rather than strict classification tasks without fine-tuning.

5.12. 2. Noisy Images

Results:

- EfficientNet: 49.89%
- ViT: 81.08%
- CLIP-ViT: 7.26%

Analysis: In the presence of noise, ViT maintained relatively strong performance with an accuracy of 81.08%, indicating its robustness to local perturbations. EfficientNet saw a drop in accuracy to 49.89%, suggesting that it is more sensitive to noise compared to ViT, which may be due to the way it extracts features. CLIP-ViT continued to perform poorly with 7.26%, reinforcing the idea that the model is not primarily designed for direct classification tasks without further adjustments or fine-tuning.

5.13. 3. Styled Images

Results:

- EfficientNet: 62.55%
- ViT: 95.33%
- CLIP-ViT: 26.83%

Analysis: The accuracies for EfficientNet and ViT on styled images remained unchanged from the original images, indicating that their performance is stable against changes in global style. CLIP-ViT improved to 26.83%, suggesting that the model's architecture may handle style variations better than it does noisy images, likely due to its focus on understanding the content in relation to text prompts.

5.14. 4. Scrambled Images

Results:

- EfficientNet: 9.25%
- ViT: 17.25%
- CLIP-ViT: 10.95%

Analysis: All models performed poorly on the scrambled images, indicating that they struggled to extract meaningful information when the spatial relationships between pixels were disrupted. ViT achieved the highest accuracy at 17.25%, which is still quite low. This suggests that even with its strong performance on original images, it lacks robustness against severe structural alterations like scrambling. EfficientNet and CLIP-ViT also had low accuracies, indicating that traditional CNN architectures and CLIP-based architectures do not perform well under these circumstances.

5.14.1. Overall Analysis and Reasoning

Model Robustness: ViT demonstrates remarkable robustness to noise and retains high accuracy across original and styled datasets. This suggests that the self-attention mechanism effectively captures important features regardless of local changes. EfficientNet shows decent performance under original and styled conditions but is more sensitive to noise and scrambled images, indicating potential limitations in its feature extraction capabilities when faced with perturbations.

CLIP-ViT Performance: The poor performance of CLIP-ViT on direct classification tasks (especially on original and noisy images) suggests it is not well-suited for classification without additional fine-tuning. Its architecture is primarily focused on image-text relationships, which explains its relatively better performance on styled images where content interpretation can leverage textual prompts.

Effects of Data Manipulation: The results indicate that models can handle transformations like style changes better than noise or scrambling. Noise injection creates local distortions that can mislead models, while scrambling disrupts spatial coherence entirely, making it challenging for all models to perform well.

In conclusion, the ViT model excels in various conditions, particularly original and noisy images, making it the best performer in this analysis. EfficientNet maintains reasonable performance but shows limitations against noise and scrambling. CLIP-ViT, while effective for tasks related to image-text relationships, requires further adaptation for pure classification tasks. Overall, the choice of model should consider the specific nature of the task and the data transformations being applied.

5.15. Task 6

5.15.1. Depthwise Convolution

5.16. Analysis of the Feature Maps

The result of the Depthwise Convolution layer shows three feature maps, each representing a different filter applied to the input image:

First Feature Map (Left)

- This map preserves significant details of the overall shape of the bird (the foreground object), and the background is less emphasized.
- It captures broader details, including the edges and general contours of the bird.
- The bird appears more prominent, indicating that the depthwise convolution has focused on the main structure of the object.

Second Feature Map (Middle)

- The middle feature map focuses more on fine-grained texture details, particularly around the surface of the bird and some elements of the background.
- Textures and local patterns in the image are more clearly visible.
- There are several patches where edges are enhanced, and some of the finer textures of the background are also visible.

Third Feature Map (Right)

- The third feature map appears to enhance the background details and some broader structures of the bird.
- It shows the bird's outline but emphasizes more subtle edges.
- The background, though slightly noisy, appears with clearer edges in this feature map, indicating that the depthwise convolution is capturing broader spatial relationships.

5.16.1. Key Takeaways

- The depthwise convolution layer effectively captures local features such as edges, textures, and contours, making it computationally efficient while maintaining important spatial details.
- The feature maps highlight different aspects of the image, ranging from broader contours to finer textures.
- However, depthwise convolution lacks the ability to capture global context, as it focuses primarily on local spatial relationships.
- This makes it efficient for extracting local features but limited in its capacity to capture broader patterns in the image.

5.16.2. Self-Attention

5.17. Analysis of the Feature Maps

The feature maps from the Self-Attention layer exhibit different characteristics compared to the depthwise convolution maps:

5.17.1. First Feature Map (Left)

- The bird is clearly visible, with attention being distributed over both the bird and parts of the background.
- The foreground object (bird) is still emphasized, but the attention seems to capture broader patterns, highlighting some areas of the background related to the object.

- The edges are not as sharply defined as in the depthwise convolution, suggesting that the model is focusing on overall structure rather than fine-grained local details.

5.17.2. Second Feature Map (Middle)

- This map shows a strong focus on the global context of the image, with the bird and background being treated almost equally in terms of importance.
- The feature map highlights the entire object (the bird) and some of the background trees, capturing relationships between distant parts of the image.
- The broader scope of attention shows how the self-attention layer captures both local and global relationships.

5.17.3. Third Feature Map (Right)

- Similar to the second map, this feature map distributes attention across both the foreground and background.
- It emphasizes the bird's body even more and highlights some distant regions of the background, indicating that the attention mechanism is considering broader spatial relationships.

5.18. Key Takeaways

- The self-attention layer excels at capturing global relationships in the image, connecting distant parts of the scene, and distributing focus more evenly between the foreground and background.
- It smooths out local details like edges, providing a more holistic view of the image.
- While it may miss finer local features that convolution captures, self-attention effectively handles complex images where global context is important.
- Combining self-attention with convolution could offer a balanced approach to feature extraction.

6. Pre-Normalization

6.1. Analysis of Feature Maps

6.1.1. First Feature Map (Left)

- The bird's shape and background are both visible, though the details are somewhat blurred. The pre-normalization has ensured that the input to the convolutional and attention layers is uniform, but the result is a slight reduction in sharpness, likely due to the smoothing effect of normalization.
- The overall contours and structure of the bird are preserved, but fine details are lost.

6.1.2. Second Feature Map (Middle)

- This feature map focuses more on the global structure, with a noticeable balance between the bird and its background. There is a broad representation of the scene, indicating that self-attention is capturing the relationships between the bird and the environment, though it is more blurred compared to the post-normalization map.
- The attention weights may have been influenced by the pre-normalized input, leading to smoother transitions between the bird and background.

6.1.3. Third Feature Map (Right)

- The third map appears to highlight background textures more prominently. This could indicate that the pre-normalized input allowed the attention mechanism to distribute focus more evenly across the entire scene, rather than just on the primary object (the bird).
- The overall image is smooth but retains some texture details in the background, albeit with less sharpness compared to post-normalization.

6.2. Key Takeaways

- Pre-normalization results in smoother feature maps by providing consistent input to the layers, which helps stabilize feature extraction.
- However, it reduces sharpness and fine-grained details compared to post-normalization.
- The model captures both local and global features, with an emphasis on global relationships, though this comes at the cost of some detail loss.

7. Post-Normalization

7.1. Analysis of Feature Maps

7.1.1. First Feature Map (Left)

- This map shows a more blurred and less distinct image. It seems to emphasize broader regions of the image, particularly the background and the edges of the bird. This suggests that the global relationships captured by the self-attention mechanism are being emphasized here.
- The normalization has a smoothing effect, which may have led to the reduction in sharpness.

7.1.2. Second Feature Map (Middle)

- This feature map shows better preservation of the bird's shape and some background details. It captures more of the contours and boundaries of the bird, suggesting that the convolutional layers are contributing local features to this map.

- The balanced focus between the foreground (bird) and the background indicates that both local and global features are contributing effectively.

7.1.3. Third Feature Map (Right)

- This map seems to capture fine-grained textures and patterns, particularly in the background. The attention mechanism may have distributed importance across small details, while the normalization helped maintain stability in how these details are represented.
- The result is textured details being emphasized without overwhelming the image with noise.

7.2. Key Takeaways

- The post-normalization model provides a balanced combination of local and global features, ensuring stable feature extraction through normalization.
- The model captures broad structures and textures while maintaining details, though some sharpness may be lost due to the smoothing effect of the normalization.
- This approach ensures stability in learning and provides a robust representation of both foreground and background details.

8. Attention Modulated

8.1. Analysis of Feature Maps

8.1.1. First Feature Map (Left)

- The bird and its surrounding environment are well-represented with soft transitions between the foreground (bird) and background. This indicates that the model is capturing a balanced mixture of both local features (edges, contours) and global features (relationships between the bird and background).
- There is some loss of sharpness, which is expected due to the smoothing effect of normalization after feature extraction.

8.1.2. Second Feature Map (Middle)

- The second map highlights global structural features, with the bird's body being emphasized, while the background has some softer, blurred regions. This suggests that the model is paying attention to the overall structure and relationships between the bird and the surrounding area.
- The foreground-background relationship is still preserved but more blended, likely due to the normalization step.

8.1.3. Third Feature Map (Right)

- The third map shows darker textures and emphasizes finer details in the background. While the foreground (bird) is still visible, the normalization smooths out some of the sharp edges, making the overall image less distinct.
- This map captures more detailed background textures than the other two, indicating that the normalization step is balancing the contribution of both the local and global features.

8.2. Key Takeaways

- The attention-modulated, provides a balanced approach to capturing both local and global features, with a smoothing effect that refines the feature maps.
- While some sharpness is sacrificed, the normalization ensures stability and consistency across the image, allowing the model to effectively represent both foreground objects and background textures.
- The result is a robust and stable feature extraction process with refined global and local context.

9. Convolution Modulated

9.1. Analysis of Feature Maps

9.1.1. First Feature Map (Left)

- The first map shows a balance between the foreground object (bird) and the background textures. While the bird's shape is distinct, the background details are also highlighted, indicating that the attention mechanism is distributing focus between different regions.
- This map captures the broader relationships between the foreground and the background, blending global and local features effectively.

9.1.2. Second Feature Map (Middle)

- The second map emphasizes global structure more clearly. The bird is represented prominently, with smooth transitions into the background. The convolution process has captured some local details, while attention has focused on broader aspects, allowing the model to see the full scene.
- The bird's overall shape is well-preserved, and the surrounding environment is softly blended.

9.1.3. THIRD FEATURE MAP (RIGHT)

- The third map focuses on fine details in both the bird and the background. Some sharper edges around the bird are visible, indicating that the convolutional features were successfully enhanced through attention modulation.
- There is a good representation of textures in both the foreground and background, with attention helping balance the global and local features.

9.2. Key Takeaways

- The convolution-modulated attention model strikes a balance between local feature extraction and global context by using convolution to capture fine details and attention to enhance these features.
- The resulting feature maps preserve sharpness and detail, while providing a holistic view of the entire image, making it effective for representing both foreground objects and background textures.

10. Contributions

M.Yaseen Shahid attempted the Task 0, 2, 5 Rana Abdul Wahab attempted Task 1,3,6 Abdullah Aamer attempted Task 4 and Re0port Writing

References

[1] R. Geirhos, P. Rubisch, C. Michaelis, et al., “ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness,” ICLR, 2019. [Online]. Available: https://arxiv.org/abs/1811.12231

[2] J. Tian, Y. Hsu, Y. Shen, H. Jin, Z. Kira, “Exploring Covariate and Concept Shift for Detection and Confidence Calibration of Out-of-Distribution Data,” 2021. [Online]. Available: https://arxiv.org/abs/2110.15231