ANN Main Project Report

Tengyang Zheng Jinpeng Zhao Zhongyan Zheng
Sun Yat-Sen University

Abstract

Building our model by applying relevant knowledge from the Artificial Neural Networks course, and continuously improving and refining the model through completing multiple tasks, in order to achieve better training results.

1. Introduction

In this study, we trained neural networks on the CUB200 Bird Dataset and Stanford Dogs Dataset to perform image classification tasks. The objective of our experiment was to achieve a classification accuracy of 90% on both datasets.

To improve the performance of the models, we implemented various training tricks, utilized transfer learning by fine-tuning pretrained models, explored the effectiveness of attending to local regions through object localization or segmentation techniques. Moreover, we investigate the use of synthetic image generation as part of data augmentation to increase the diversity of the training data,conducte a comparison between ViT and CNN, have also dedicated significant efforts to improving the interpretability of our model and improve the robustness of model.We have also been experimenting with applying self-supervised learning to our model.Through extensive experimentation and analysis, we observed significant improvements in the accuracy of our models. Our findings highlight the effectiveness of these strategies in enhancing the performance of neural networks for image classification tasks.

The source code can be found on the project page on https://github.com/sumuzhe317/ANN-FinalProject.git

2. Related Work

ResNeXt-101 [9] ResNeXt-101 is a deep neural network model based on the principles of Residual Networks (ResNet) and Grouped Convolution. It was introduced in the paper "Aggregated Residual Transformations for Deep Neural Networks".

ResNeXt-101 improves upon ResNet by combining the strengths of both Inception and ResNet. It enhances the network's representation and learning capabilities by introducing grouped convolution and aggregated residual connections.

In our project, we use the resnext101-32x8d model, where "32" indicates that each residual block has 32 groups (cardinality is 32), and "8d" represents the expansion factor of the bottleneck structure.

Here is a summary of the principles and innovations of the ResNeXt-101 model:

Grouped Convolution: ResNeXt-101 introduces the concept of grouped convolution. It involves dividing the input feature map into multiple groups and performing independent convolution operations on each group. The output feature maps from each group are then concatenated. By increasing the number of groups, the model can better capture the correlations between features, thereby enhancing the expressive power of the features. Grouped convolution allows the network to simultaneously utilize multiple branches to learn features, leading to improved performance. This design increases the network's ability to perform nonlinear transformations and improves its ability to model complex features.

Aggregated Residual Connections: ResNeXt-101 adopts the strategy of aggregated residual connections. Instead of simply adding the feature maps obtained from grouped convolutions, they are concatenated together. This aggregation method increases the diversity of features and allows the network to gather and fuse information from multiple paths, resulting in a more comprehensive representation of features in the image. Aggregated residual connections contribute to enhancing the network's representation capabilities and make the network more flexible and adjustable.

In the upcoming tasks, we will be using this model as the foundation for our experiments.

3. Method

3.1. Task 1-2

Due to the close relevance and foundational nature of Task 1 and Task 2, we have decided to merge their implementations in the project. Furthermore, the subsequent tasks will also build upon the foundation established in Task 1-2 for further implementation.

Task 1: Various training tricks to improve model performance

In image classification tasks, to prevent overfitting during model training caused by limited sample data, data augmentation techniques can be employed to improve model performance. In Task 1, we experimented with and adopted various data augmentation methods, including random image grayscale transformation, random horizontal flipping, data resizing, image padding, and normalization. These techniques were used to enhance the performance of our model.

We also conducted comparative tests on different loss functions, optimizers, and learning rate adjustment strategies for the model. We adjusted the corresponding parameters and ultimately selected the combination that yielded better performance. Specifically, we used the cross-entropy loss function and stochastic gradient descent (SGD) optimizer, along with the stepLR learning rate adjustment strategy.

The cross-entropy loss function is widely used in classification tasks, especially in multi-class classification problems. It evaluates the performance of a model by measuring the difference between its predicted outputs and the true labels. The cross-entropy loss effectively penalizes classification errors and encourages the model to learn in the right direction. It possesses good mathematical properties for evaluating the accuracy of model predictions in classification tasks. SGD is a commonly used optimization algorithm for updating model parameters during the training process. SGD computes the gradients of the loss function on each training sample and updates the model parameters in the direction of the negative gradient, gradually reducing the value of the loss function. SGD has the advantages of simplicity and ease of implementation, and it exhibits good convergence performance on large-scale datasets. The stepLR strategy is a learning rate adjustment technique used to dynamically adjust the learning rate during training. The learning rate is a crucial hyperparameter that controls the step size of parameter updates in a model. An appropriate learning rate can accelerate the convergence speed and improve the performance of the model. The stepLR strategy adjusts the learning rate according to a predefined decay rate after each predetermined step (epoch) of training. By gradually decreasing the learning rate, the model can approach the optimal solution more stably during the later stages of training, enhancing its generalization ability.

The stepLR learning rate adjustment strategy provides flexibility in dynamically adjusting the learning rate during the training process, thereby improving the model's performance at different stages. The combination of these choices can effectively optimize the model's performance during training, enabling it to better fit the training data and achieve good generalization on test data.

In addition to preprocessing the image data and setting the model parameters, we visualized the training results using TensorBoard. By visualizing the model's loss as a chart, we could evaluate the training effectiveness more conveniently. This approach allowed us to perform multiple comparisons and tests to identify the optimal parameter configuration for better training results. Furthermore, it helped us determine if the model training was converging in subsequent training tasks, thereby improving training efficiency.

Task 2: Transfer learning: fine-tune pretrained model

In Task 2, we experimented with loading different pretrained weights and compared the training performance with the baseline weights loaded from PyTorch. After evaluation, we ultimately decided to use the pre-trained weights from torchvision, specifically the ResNeXt101-32X8D weights, as they exhibited better training results. The results are shown in Table 1.

Model	Dateset	Accuracy
Baseline	bird	13.24%
Improved model	bird	83.29%
Baseline	dog	23.72%
Improved model	dog	92.39%

Table 1. Comparison of Accuracy between Baseline and Improved model

3.2. Task 3: Attend to local regions: object localization or segmentation

We attempted to use MMDetection [2] for object detection on both the training and testing datasets, aiming to use the object detection results obtained from MMDetection as inputs for our model.

MMDetection is a deep learning-based multimodal object detection toolbox specifically designed for object detection tasks in images and videos. It provides a range of efficient and powerful algorithms and models that can be applied in various application areas such as intelligent security, autonomous driving, and human-computer interaction. Developed based on the PyTorch framework, MMDetection features a modular design and flexible configuration options, allowing users to customize and extend models according to their specific needs. It offers a rich model library, including Faster R-CNN, Cascade R-CNN, RetinaNet, etc., which have been extensively experimented and optimized to achieve good detection performance in different tasks and scenarios.

By adjusting the parameter configuration files of MMDetection, We utilized neural networks from the mmdetection framework to perform object detection on our dataset. Following the training, we employed the trained neural network to perform object detection on images and extracted the bounding box with the highest confidence score. These extracted bounding boxes were then cropped to generate a new

dataset. However, the results of the model training showed a slight decrease in accuracy compared to task1-2.The results are shown in Table 2.

The possible reason for this could be related to feature representation. If the feature representation in the object detection results significantly differs from the feature representation that the training model can handle, the model may struggle to learn effective information from these features.

We believe that crop classification only provides positional information of the targets, but it may result in varied input sizes for the model and may lack other important features such as scale, orientation, and texture. Therefore, we further attempted square cropping (squreCropClassification), ensuring consistent input sizes for the model and preserving some crucial features of the original image. On the bird dataset, there was a slight improvement in accuracy compared to crop classification, while on the dog dataset, the accuracy decreased slightly.

Model	Dateset	Accuracy
task1-2	bird	83.29%
with Crop	bird	83.52%
wirh squreCrop	bird	82.29%
task1-2	dog	92.39%
with Crop	dog	90.7%
wirh squreCrop	dog	90.71%

Table 2. Comparison of Accuracy between Baseline and MMdetection with Crop

3.3. Task 4: Synthetic image generation as part of data augmentation

Synthetic image generation as part of data augmentation is a common technique that can help improve the performance of machine learning models for image processing tasks. This approach utilizes generative models such as Generative Adversarial Networks (GANs) or Variational Autoencoders (VAEs [7]) to generate new synthetic images based on existing real images. By generating new samples from the training set images, generative models effectively expand the scale of the training set.

In this task, we use DCGAN [8] to generate synthetic images as part of the dataset.

DCGAN (Deep Convolutional Generative Adversarial Networks) is a generative adversarial network (GAN) model based on deep convolutional neural networks. DCGAN combines the advantages of deep convolutional neural networks and GANs, enabling the generation of realistic images. Its innovation lies in the integration of the architecture and techniques of convolutional neural networks to overcome some of the challenges faced by traditional GAN models in image generation tasks. The generator and discriminator networks in DCGAN are deep neural networks composed of convolutional layers, batch normalization layers, activation functions, and other components. The generator network takes a low-dimensional latent space vector as input and gradually transforms it into an output image that resembles the images in the training set, using transpose convolutional layers (also known as deconvolutional layers) instead of traditional fully connected layers. This helps to preserve the spatial structure and detailed features of the generated images.

We also tried using the StyleGAN [6] architecture for generating synthetic images, but training StyleGAN proved to be challenging, requiring significant time and computational resources. Due to these limitations, we opted for DCGAN instead. While DCGAN may not achieve the same performance as StyleGAN, it has a shorter training time. Due to time and resource constraints, we adjusted the size of the generated images to 32x32 and then applied upsampling, as well as experimented with using super-resolution networks to enhance the generated images.

The comparison between the images generated by DCGAN and the original dataset is shown in Fig. 1.

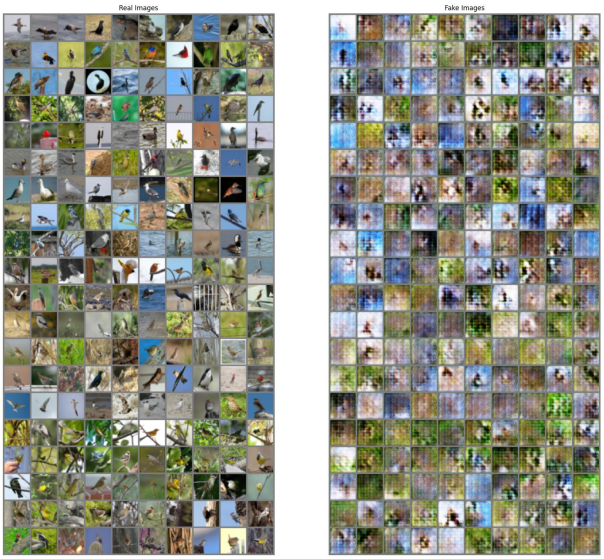


Figure 1. The original dataset and images generated by DCGAN.

We added the generated images from DCGAN as a part of the training set for model training, and the experimental results indicate that the model performs better on the test set compared to task 1-2,the result is shown in Table 3. However, due to the unstable training of DCGAN, the loss function of the model oscillates repeatedly, and the generated images may lack some details, clarity, and realism. Therefore, in the future, we plan to utilize more advanced GAN models. We believe that using these updated GAN models will further enhance the performance of the model.

Model	Dateset	Accuracy
task1-2	bird	83.29%
use DCGAN	bird	84.54%
task1-2	dog	92.39%
use DCGAN	dog	92.79%

Table 3. Comparison of Accuracy between Baseline and using DCGAN

3.4. Task 5: ViT model backbone vs. CNN backbone: explore how to effectively use ViT

ViT (Vision Transformer) and CNN (Convolutional Neural Network) are two popular types of neural network architectures commonly used in computer vision tasks.

The ViT (Vision Transformer) model was introduced by the Google team in their paper [4] and the aim of the ViT model is to directly apply Transformer models to computer vision classification tasks with minimal modifications.

The core idea of ViT is to convert input images into a series of learnable vector representations suitable for Transformers. The original image is divided into fixed-size patches, and each patch contains a vector representation. These vectors are flattened and passed through Transformer encoders for processing.

One significant advantage of ViT is its ability to capture global contextual information without relying on local convolution operations [1]. This makes ViT advantageous for visual tasks that involve large-sized images and rely on global structures, such as image classification and object detection.CNNs progressively expand the convolutional window by convolving layer by layer, capturing a wider range of information. On the other hand, the ViT model, even at the lowest level, can leverage self-attention mechanism to allow for a larger receptive field.

On the other hand,ViT treats the image as a sequence of non-overlapping patches. These patches are flattened and linearly projected into a lower-dimensional latent space. Then, ViT applies self-attention mechanisms to capture relationships between the patches.CNNs use convolutional operations to process images. Convolutional filters are applied to extract local features from different regions of the image. Max pooling or average pooling is used for downsampling the spatial dimensions, reducing complexity, and extracting important information.

The objective of this task was to compare the performance differences between Vision Transformer (ViT) and Convolutional Neural Network (CNN) by designing and implementing a series of experiments. We evaluated the performance of both neural network architectures on an image classification task. The results indicated significant differences between ViT and CNN in our model.

We used the bird and dog datasets for image classification as our experimental benchmarks. We constructed ViT and CNN models on the same training and validation sets. For the ViT model, we utilized the vit framework from PyTorch. And then we compared their performances.

We found that ViT exhibited higher accuracy in our model. It was able to better capture global features and contextual information in images, resulting in more precise classification results. Despite ViT's impressive accuracy, we observed that it had larger model parameters, occupying more memory space. In comparison, CNN had smaller model size and parameter count. Additionally, ViT often required longer training time, especially on relatively larger datasets. In contrast, CNN typically converged faster.

Model	Dateset	Accuracy
task1-2	bird	83.29%
use ViT	bird	84.13%
task1-2	dog	92.39%
use ViT	dog	92.47%

Table 4. Comparison of Accuracy between Baseline and using ViT

Our experimental results demonstrated that ViT outperformed CNN in our model for the image classification task. This aligns with ViT's global perception capabilities and self-attention mechanism. However, CNN had advantages in terms of model parameter efficiency and training time.

3.5. Task 6: Interpretation of the model: visualization of model predictions

Complex neural networks are often considered as "black boxes" to humans because they consist of a large number of parameters and layers, making it difficult to directly understand their internal workings. We can only observe their output results on input data without gaining deep insights into how the network makes decisions and learns. However, it is precisely due to their complexity that these neural networks are capable of tackling various challenging tasks and achieving significant advancements.

To address the interpretability issue of neural networks, visualizing the model becomes crucial. Through visualization techniques, we can attempt to reveal the internal mechanisms of neural networks, making their decision-making process more transparent and understandable. Visualization tools such as heatmaps, activation maps, and feature visualizations can provide intuitive representations of the model's behavior. Heatmaps, in particular, are a visualization tool commonly used in image processing and computer vision tasks to indicate the saliency of different regions in an image with respect to a target or region of interest. They help us understand the decision process and focus areas of neural networks in tasks like image classification, object detection, and image segmentation.

Heatmaps can aid in visualizing the feature maps of intermediate layers in a neural network, enabling a better understanding of how the network processes and extracts features from input data. By observing the heatmaps, we can intuitively grasp which regions contribute significantly to the image classification problem. Furthermore, heatmaps can assist in debugging and diagnosing network models. By examining the heatmaps, we can identify potential issues within the feature maps, such as gradient vanishing or exploding, and subsequently refine the network structure or adjust hyperparameters.

In Task 6, we visualized the model training process and displayed the feature maps of each channel as heatmaps during network inference. The results are shown in Fig. 2.

It can be observed that as we move up the hierarchy of convolutions, the visualized features become increasingly blurry, indicating the extraction of more abstract semantic information.

3.6. Task 7: Robustness of the model: adversarial examples as input, improve robustness

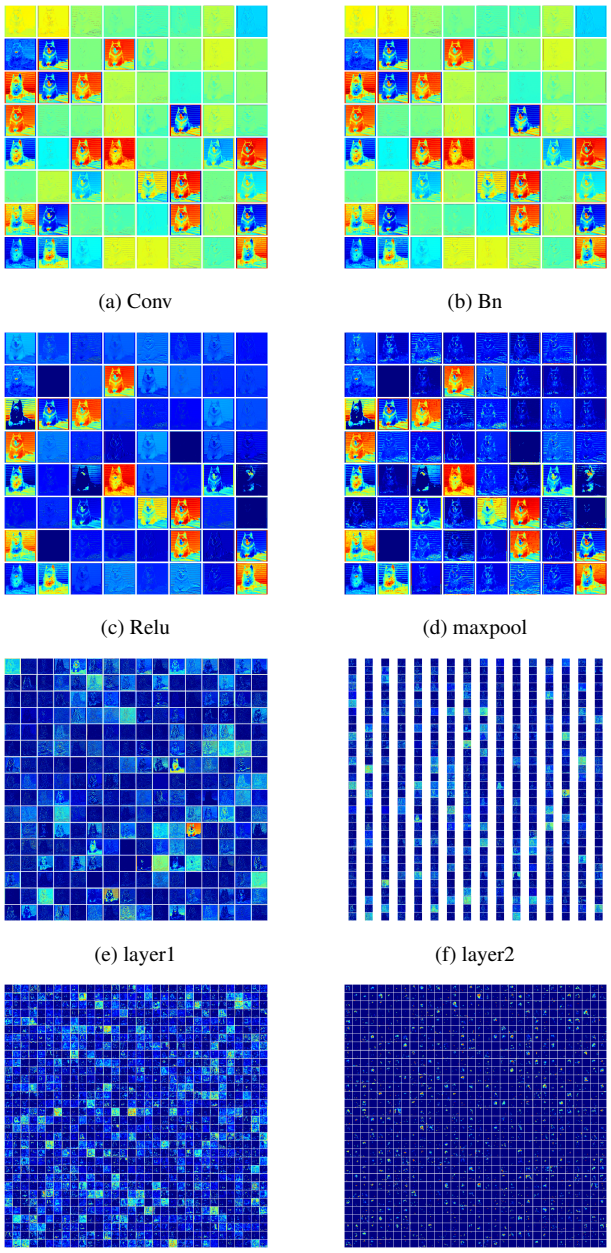
FGSM [5](Fast Gradient Sign Method) is an algorithm for generating adversarial examples based on gradients. It belongs to the category of non-targeted attacks in adversarial attacks, which means that it does not require the adversarial examples to be predicted as a specific target class, but rather aims to make the prediction different from the original sample's prediction. Essentially, FGSM is a gradient ascent algorithm that maximizes the loss function by making slight modifications to the input, resulting in a large discrepancy between the predicted output and the actual value.

The FGSM attack method utilizes the gradient information of the input sample to generate adversarial examples. Specifically, it calculates the gradient of the input sample with respect to the loss function and adjusts the pixel values of the input sample based on the sign of the gradient. The magnitude of this adjustment is controlled by a hyperparameter called the perturbation size (epsilon), which determines the degree of difference between the adversarial example and the original sample.

The key idea of FGSM is to maximize the value of the loss function while keeping the distance between the adversarial example and the original sample as small as possible, leading to misclassification by the model. Due to its utilization of only the gradient information of the input sample, FGSM is computationally efficient and does not require retraining the model. This makes FGSM a fast, simple, yet effective attack method. By generating adversarial examples, FGSM can reveal the model's instability when facing minor perturbations, providing insights into the model's vulnerabilities and weaknesses.

In task7, we used FGSM to generate adversarial examples with a probability of 0.5 during the training process. This means that approximately half of the samples in the training data were adversarial examples. The results are shown in Table 5.

The use of FGSM to generate adversarial examples did not have a significant impact on the accuracy. This could be because FGSM is a relatively simple adversarial attack method, which the model may find it easier to understand. Another possibility is that the chosen perturbation size was too small, resulting in adversarial examples that closely re-



(a) Conv (b) Bn (c) Relu (d) maxpool (e) layer1 (f) layer2 (g) layer3 (h) layer4

Figure 2. Heatmaps

Model	Dateset	Accuracy
task1-2	bird	83.29%
use FSGM	bird	84.10%
task1-2	dog	92.39%
use FSGM	dog	92.00%

Table 5. Comparison of Accuracy between Baseline and using FSGM

semble the original samples. As a result, the model may have difficulty distinguishing between adversarial examples and original samples, leading to less noticeable effects in the adversarial training process.

3.7. Task 8: Self-supervised learning

SimCLR [3](A Simple Framework for Contrastive Learning of Visual Representations) is a contrastive learning network that can train and infer from datasets with limited labels. It consists of two parts: unsupervised learning and supervised learning. SimCLR is a simple framework for contrastive learning of visual representations, proposed in the context of self-supervised learning, with the aim of learning meaningful visual feature representations by maximizing the similarity between image pairs from different perspectives.

The core idea of SimCLR is to train the model by maximizing the similarity between positive sample pairs and minimizing the similarity between negative sample pairs. Specifically, SimCLR utilizes a contrastive loss function to achieve this objective. The contrastive loss function calculates the similarity measure between positive and negative sample pairs, increasing the similarity between positive pairs and decreasing the similarity between negative pairs.

In lab8, we use SimCLR for training in two stages. In the first stage, we perform unsupervised learning by applying two rounds of random image augmentations to the input images. Each augmented image is passed through the network for training, and the loss is computed and gradients are updated. In the second stage, we load the trained parameters of the feature extraction layer from the first stage and perform supervised learning using a small labeled dataset (training only the fully connected layer). The loss function in this stage is the CrossEntropyLoss.The results are shown in Table 6.

Model	Dateset	Accuracy
task1-2	bird	83.29%
use SimCLR	bird	41.42%
task1-2	dog	92.39%
use SimSLR	dog	57.11%

Table 6. Comparison of Accuracy between Baseline and using SimSLR

The poor performance of the model could be attributed to the limited training time. It's possible that the model was not trained to convergence, resulting in suboptimal training results and a noticeable decrease in accuracy. This limitation might have prevented the model from fully showcasing the advantages of SimSLR.

4. Conclusion

By combining multiple strategies, including training tricks, transfer learning, attention mechanisms, data augmentation, ViT and CNN backbones comparison, model interpretation, robustness evaluation, and self-supervised learning, our objective was to build a robust and accurate model for fine-grained visual classification tasks.

References

[1] Nikolas Adaloglou. Transformers in computer vision. *https://theaisummer.com/*, 2021. 4

[2] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. Mmdetection: Open mmlab detection toolbox and benchmark. *CoRR*, abs/1906.07155, 2019. 2

[3] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 1597–1607. PMLR, 13–18 Jul 2020. 6

[4] 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. *CoRR*, abs/2010.11929, 2020. 4

[5] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples, 2015. 5

[6] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. *CoRR*, abs/1812.04948, 2018. 3

[7] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv.org*, 2014. 3

[8] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *Computer ence*, 2015. 3

[9] Saining Xie, Ross Girshick, Piotr Dollar, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 1