Project Report

Hadas Abraham , Dor Malka

July 2023

1 Introduction

1.1 The Goal of The Project

The goal of this project is to develop a deep-learning model capable of accurately detecting and identifying Pokémon characters in images. By leveraging the power of convolutional neural networks (CNNs), the model will learn to recognize unique visual patterns and features associated with different Pokémon species, enabling it to accurately classify and locate Pokémon within images. Furthermore, after classification, our network will produce and show on-the-fly some interesting and fun facts about the Pokémon.

1.2 Motivation

1. Popularity and Nostalgia: Pokemon is a globally recognized and beloved franchise that has captured the hearts of millions of people since its inception. By developing a Pokemon classifier, you can tap into the nostalgia and excitement surrounding these iconic creatures, making it an engaging and relatable project for both you and your classmates.

2. Real-World Application: Deep learning classifiers have numerous practical applications, and developing a Pokemon classifier allows you to showcase the potential of deep learning techniques in image recognition tasks. You can highlight how these models can be used to accurately classify and identify different Pokemon species based on their visual attributes, demonstrating the broader applicability of deep learning in image analysis.

3. Interpretability and Visualization: Pokemon classifiers can offer exciting opportunities for interpreting and visualizing deep learning models. You can analyze the learned features, visualize activation maps, or even create interactive visualizations to demonstrate how the model recognizes different Pokemon attributes. This not only enhances your understanding of deep learning techniques but also allows you to present your findings in an engaging and visually appealing manner.

2 Method

Our decision to examine both Transfer Learning and Self-Supervised Learning techniques for this project stems from their respective strengths and potential benefits. Transfer Learning allows us to leverage pre-trained models and their learned features, enabling faster and more efficient training on our Pokemon dataset. On the other hand, exploring Self-Supervised Learning techniques offers the opportunity to train models in an unsupervised manner, extracting meaningful representations from the data itself, which can be advantageous for the diverse and visually intricate nature of Pokemon species. By investigating both approaches, we aim to assess their effectiveness and determine the most suitable method for accurate and robust Pokemon classification.

2.1 The Algorithms

2.1.1 Transfer Learning

We selected the VGG (Visual Geometry Group) pre-trained model for our transfer learning approach in the Pokemon project due to its strong performance in image classification tasks. VGG is known for its deep architecture, which allows it to capture intricate visual features and patterns, making it well-suited for recognizing the complex attributes of different Pokemon species. By leveraging the pre-trained weights of VGG, we can benefit from its extensive knowledge learned from a large-scale dataset, saving time and computational resources while achieving competitive classification accuracy for our Pokemon dataset.

2.1.2 Self-Supervised Learning

We chose to explore self-supervised learning approaches such as BYOL, DINO, SimCLR, and SimSiam for our Pokemon project due to the limited size of our dataset and the presence of visually similar Pokemon species. These self-supervised learning methods excel at learning representations from unlabeled data, enabling the models to capture subtle differences between visually similar species and generalize well to unseen examples. By leveraging the self-supervised learning algorithms' ability to learn meaningful representations, we aim to overcome the challenges posed by the limited dataset and accurately classify Pokemon species based on their visual attributes.

2.2 The Architecture

2.2.1 VGG - Transfer Learning

The architecture used for transfer learning with the VGG model involves loading the pre-trained VGG16 model. We choose the VGG model for its deep architecture and good performance on various computer vision tasks [1]. We set the last fully connected layer of the classifier to have the number of classes in our dataset, which allows the model to adapt to our specific classification task. The input size is set to 614, reflecting the size of the input features. We utilize the CrossEntropyLoss as the loss function and optimize the parameters using the SGD optimizer with a learning rate of 0.001 and momentum of 0.9.

2.2.2 Self-Supervised Learning Architecture

The self-supervised learning architecture consists of a backbone network, a projection head, and a prediction head. The backbone network, such as ResNet18 in this case, extracts high-level features from the input data. The projection head maps the extracted features to a lower-dimensional space, while the prediction head further transforms the projected features. The model aims to maximize the similarity between differently augmented views of the same input.

In our model, after pre-training the self-supervised model, we utilize the cross-entropy loss function for fine-tuning. For the fine-tuning process, we optimize all parameters, including the ones from the added classification layers. To optimize the parameters, we use the Adam optimizer with a learning rate of 0.001.

2.2.3 SimCLR

In our case, we have chosen ResNet18 as the backbone network for feature extraction. The SimCLRProjectionHead is then applied to the flattened features to obtain the projected representation. We have set the input size to 32 and disabled Gaussian blur in the SimCLRTransform [2]. The chosen batch size is 256, which affects the number of samples processed in each training iteration. The NTXentLoss criterion is used to measure the agreement between augmented views of the same input. The optimizer chosen is Stochastic Gradient Descent (SGD) with a learning rate of 0.06, which controls the step size during gradient descent updates.

These choices were made based on the existing literature and empirical results. SGD is a commonly used optimizer in self-supervised learning, and a higher learning rate of 0.06 can promote faster convergence. The batch size of 256 strikes a balance between computational efficiency and effective representation learning.

The modified SimCLR architecture in our code includes additional layers for classification purposes. We replace the last layer of the projection head with a fully connected layer that outputs the desired number of classes. Additionally, a BatchNorm1d layer is added to normalize the features. The input size for the classification part is set to 614, reflecting the size of the input features after passing through the projection head. These modifications enable the SimCLR model to be used as a classifier for our specific task of classifying Pokemon species.

2.2.4 SimSiam

In our case, we have chosen ResNet18 as the backbone network for feature extraction. The SimSiamProjectionHead and SimSiamPredictionHead are applied to the flattened features to obtain the projected and predicted representations [3], respectively. We have set the input size to 32 for transformation. The chosen batch size is 256, which determines the number of samples processed in each training iteration. The NegativeCosineSimilarity criterion measures the negative cosine similarity between the projected and predicted representations. The optimizer chosen is Adam with a learning rate of 0.005.

These choices were made based on the existing literature and empirical results. Adam optimizer is commonly used for self-supervised learning due to its adaptive learning rate mechanism. The chosen learning rate of 0.005 balances between convergence speed and stability. The batch size of 256 was selected considering computational efficiency and effective representation learning.

We then modify the last layer of SimSiam of the prediction head by replacing it with a fully connected layer that outputs the desired number of classes. The input size for the classification part remains the same, set to 614, which corresponds to the size of the input features after passing through the backbone and the projection head. These modifications enable the SimSiam model to be used as a classifier for our specific task of classifying Pokemon species.

2.2.5 BYOL

During training, the model has two sets of parameters: the online network and the target network. The online network (backbone, projection head, and prediction head) is updated using gradient descent, while the target network is updated by exponentially moving the parameters of the online network towards the target network. This moving average helps stabilize and improve the training process.

In the forward pass, the input data is passed through the backbone network to obtain feature embeddings. These embeddings are then projected using the projection head, and the projected features are further processed by the prediction head. The output of the prediction head is used for training [4].

The BYOL architecture is trained using a negative cosine similarity loss. The model is initialized with a ResNet18 backbone and then converted to the BYOL architecture by creating an instance of the BYOL class and passing the backbone network. The input data is transformed using the SimCLRTransform, and the model is trained using stochastic gradient descent (SGD) optimizer with a learning rate of 0.06.

2.2.6 DINO

The DINO architecture consists of a student network and a teacher network, both of which have a backbone network (e.g., ResNet) and a projection head. The student network is the main network that learns to predict the outputs of the teacher network. The teacher network is a slower-moving copy of the student network that provides targets for the student's predictions.

During training, the student network takes input data and extracts feature embeddings using the backbone network. These embeddings are then projected using the student's projection head. The student network aims to match its own predictions with the predictions of the teacher network [5].

The teacher network is initialized as a copy of the student network, but its parameters are frozen during training. It also has a projection head that operates in the same way as the student's projection head. The teacher network provides the targets for the student network's predictions.

In the forward pass, the input data is passed through the student's backbone network to obtain feature embeddings. These embeddings are then projected using the student's projection head. The output of the projection head is used for training and is compared to the output of the teacher's projection head.

The DINO architecture is trained using the DINO loss, which is a specific loss function designed for the architecture.The model is initialized with a ResNet18 backbone and the appropriate input dimension. The DINO architecture is then created by instantiating the DINO class and passing the backbone and input dimension. The input data is transformed using the DINOTransform. The model is trained using the Adam optimizer with a learning rate of 0.001.

2.3 Loss functions

2.3.1 CrossEntropyLoss

The CrossEntropyLoss function is a commonly used loss function for multi-class classification tasks. It computes the cross-entropy loss between the predicted probabilities and the true labels. The loss function calculates the negative log probability of the true class, penalizing the model for incorrect predictions and encouraging it to assign higher probabilities to the correct class.

$$\text{loss}(x, \text{class}) = -\log\left(\frac{\exp(x[\text{class}])}{\sum_j \exp(x[j])}\right) \tag{1}$$

where:

1. x is the input logits.
2. $class$ is the true class label.

2.3.2 NTXentLoss

The loss function used in SimCLR is the NTXentLoss, which stands for Normalized Temperature-scaled Cross Entropy Loss. It is a contrastive loss that encourages the model to maximize agreement between differently augmented views of the same input while minimizing agreement between views of different inputs.

The NTXentLoss essentially compares the similarity between positive pairs and negative samples after applying a temperature scaling factor. By maximizing the agreement between positive pairs and minimizing the agreement between negative samples, the model learns to capture meaningful representations that can discriminate between different views of the same input.

$$NTXentLoss = -\frac{1}{N}\sum_{i=1}^{N}\left[\frac{\exp(sim(x_i, x_j)/\tau)}{\sum_{k=1}^{N}\exp(sim(x_i, x_k)/\tau)}\right] \tag{2}$$

Where:

1. N is the batch size.
2. x_i and x_j are positive pairs (augmented views of the same input).
3. x_k represents negative samples (augmented views of different inputs).
4. $sim(a, b)$ calculates the similarity between vectors a and b, often computed as the cosine similarity.
5. τ is the temperature parameter that controls the concentration of the distribution.

2.3.3 Negative Cosine Similarity

The loss function used in SimSiam is the Negative Cosine Similarity. It is a contrastive loss that encourages the model to maximize the cosine similarity between positive pairs of representations and minimize the cosine similarity between negative pairs.

The Negative Cosine Similarity loss computes the cosine similarity between the predicted and projected representations and takes the negative of the average similarity across the batch. By maximizing the cosine similarity between positive pairs and minimizing the similarity between negative pairs, the model learns to capture meaningful representations.

$$NegativeCosineSimilarity = -\frac{1}{N}\sum_{i=1}^{N}\cos(p_i, z_i) \tag{3}$$

Where:

1. N is the batch size.
2. p_i represents the predicted representation
3. z_i represents the projected representation.
4. $cos_sim(a, b)$ calculates the cosine similarity between vectors a and b.

2.3.4 DINO Loss

The DINO loss is a specific loss function designed for the DINO (Emerging Properties in Self-Supervised Vision Transformers) architecture. It encourages the student network to match its own predictions with the predictions of the teacher network. The DINO loss combines two components: a consistency loss and a clustering loss. The consistency loss measures the similarity between the student's predictions and the teacher's predictions. The clustering loss promotes the formation of clusters in the student's embeddings.

$$\text{DINO Loss} = \lambda_1 \cdot \text{Consistency Loss} + \lambda_2 \cdot \text{Clustering Loss} \tag{4}$$

Where:

1. λ_1 and λ_2 are the weights for the consistency loss and clustering loss, respectively.
2. Consistency Loss: $\text{Consistency Loss} = -\frac{1}{N}\sum_{i=1}^{N}\frac{q_i \cdot t_i}{|q_i| \cdot |t_i|}$ q_i and t_i are the normalized student and teacher predictions for instance i, respectively.
3. N is the total number of instances.
4. Clustering Loss: $\text{Clustering Loss} = \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}\text{Sim}ij \cdot \left(\text{Sim}ij - \frac{1}{N}\sum_{k=1}^{N}\text{Sim}_{ik}\right)$
5. Sim_{ij} is the similarity between the embeddings of instances i and j.
6. N is the total number of instances.

3 experiments and results

3.1 The Dataset

We used the Pokemon dataset from the link:(https://www.kaggle.com/datasets/lantian773030/pokemonclassification) it consists of 150 classes of Pokemon from different generations. Each species is represented by approximately 30 images. This dataset poses several challenges when it comes to our goal of classification.

1. **Limited Dataset**: With only 30 images per Pokemon species, the dataset is relatively small, considering the complexity and diversity of the Pokemon universe. This limited amount of data per class can make it difficult for the models to learn comprehensive and discriminative features for accurate classification.
2. **Class Imbalance**: The dataset suffers from class imbalance, as each species is represented by an equal number of images. Some Pokemon species may have more distinct visual features or variations, making them easier to classify, while others may have less representative images. This imbalance can affect the model's ability to generalize well across all classes.
3. **Similar Species**: The Pokemon dataset contains species that are visually similar to each other. Different generations of Pokemon often have evolutionary lines with subtle differences in appearance. Distinguishing between similar species solely based on visual features can be challenging, even for humans. This similarity adds another layer of complexity for the model to accurately classify these closely related species.

3.2 The Experiments

3.2.1 VGG Experiment

In the experiment with the VGG model, we trained the model using the architecture described above for 200 epochs taking approximately 133 minutes. This approach of using transfer learning with a pre-trained VGG model proved to be time-saving compared to training the model from scratch.

Within the first 20 epochs of training, we observed that the model already achieved a 48% accuracy, indicating that the pre-trained VGG model provided a good starting point for our classification task. This early achievement of reasonable accuracy demonstrates the effectiveness of transfer learning in leveraging the learned features from a large dataset (ImageNet) and applying them to our specific task.

After completing the full 200 epochs of training, the model achieved an accuracy of approximately 70%. This improvement in accuracy over time demonstrates the model's ability to learn and adapt to the nuances of the Pokemon dataset. By utilizing the pre-trained VGG model, we were able to significantly reduce the training time and achieve satisfactory results in terms of classification accuracy.

3.2.2 SimCLR Experiment

In the experiment with the SimCLR self-supervised model, we trained the model using the architecture described above for 400 epochs in the pre-training phase and an additional 250 epochs in the fine-tuning phase. This approach of utilizing self-supervised learning with SimCLR offered time-saving benefits compared to training a model from scratch.

During the pre-training phase, the SimCLR model was trained for 400 epochs, taking approximately 364 minutes. This time investment in pre-training allowed the model to learn meaningful representations from the unlabeled data, which served as a strong foundation for subsequent fine-tuning.

In the fine-tuning phase, the pre-trained SimCLR model was further trained for 250 epochs, taking approximately 280 minutes. This process involved training the model on the labeled data with the objective of improving its performance on the specific classification task.

After the completion of the training process, the SimCLR model achieved an accuracy of approximately 30%. Although the accuracy may appear relatively low, it is important to note that self-supervised learning aims to learn useful representations rather than directly optimize for classification accuracy. The trained model can be further fine-tuned or utilized as a feature extractor in downstream tasks.

3.2.3 SimSiam Experiment

In the experiment with the SimSiam self-supervised model, we trained the model using the architecture described above for 400 epochs in the pre-training phase and an additional 200 epochs in the fine-tuning phase. The pre-training phase took approximately 299 minutes.

At the beginning of the training, we chose to prune the training algorithm. During the initial epochs, there was a significant jump in accuracy, reaching 26%. However, after this initial improvement, the accuracy remained relatively stable throughout the subsequent epochs.

After 20 epochs, we observed a decrease in accuracy, and the validation accuracy became very low. This indicates that the model may have reached a plateau or encountered difficulties in further learning. It is possible that the chosen pruning technique affected the model's ability to generalize well to the validation data.

While the experiment did not yield the desired accuracy, it provides valuable insights into the training dynamics of the SimSiam self-supervised model. Further analysis and exploration of different pruning strategies or model architectures may be required to improve the model's performance.

3.2.4 BYOL Experiment

In the experiment with the BYOL self-supervised model, we trained the model using the architecture described above for 500 epochs. The pre-training phase took approximately 363 minutes. However, we encountered a challenge during this experiment.

After the pre-training phase, we discovered that the BYOL architecture didn't defaultly save its gradients. This posed a difficulty in using the pre-trained model for fine-tuning and learning the parameters of the last

layer. Without access to the gradients from the pre-training phase, we were unable to effectively leverage the pre-trained model for further training.

This limitation hindered our ability to fully exploit the benefits of transfer learning and fine-tuning with the BYOL model. Consequently, it impacted the overall performance and effectiveness of the fine-tuning process.

3.2.5 DINO Experiment

In the experiment with the DINO self-supervised model, we initiated the pre-training phase. However, we encountered a challenge due to the time required to train each epoch. On average, each epoch took approximately 8 minutes to complete, which proved to be unrealistic given our limited computational resources.

4 conclusion

1. Transfer Learning with VGG: The experiment demonstrates the effectiveness of transfer learning with pre-trained models like VGG in reducing training time and achieving relatively high accuracy for the Pokemon classification task.

2. The wrong prediction in the VGG experiment: The challenges faced in the VGG experiment can be attributed to the inherent difficulties in accurately classifying certain Pokemon. The model itself was trained well, but the mistakes in prediction were due to specific factors such as:

   (a) *Similarity within evolution lines*: Pokemon from the same evolution line often share similar visual characteristics, making it challenging for both the model and humans to differentiate between them. This inherent similarity posed a difficulty in accurately classifying Pokemon within these specific groups.

   (b) *Pokemon with almost identical features*: Some Pokemon may have very subtle differences in their visual attributes, such as colors, shapes, or body structures. These small variations can lead to misclassifications, even for a well-trained model, as distinguishing these minute details becomes more challenging.

3. SimCLR: The experiment reveals that while SimCLR offers a promising self-supervised learning approach, achieving satisfactory accuracy requires extensive training and fine-tuning. The limited accuracy achieved suggests the need for further optimization or augmentation techniques.

4. SimSiam: The experiment highlights the challenges of training the SimSiam model, with fluctuations in accuracy and a subsequent decline in performance. This indicates the need for careful tuning of hyperparameters and addressing any underlying architectural limitations.

5. BYOL: The experiment showcases the importance of addressing implementation details in self-supervised learning models like BYOL. The inability to save gradients during the pre-training phase hindered the utilization of the pre-trained model for fine-tuning, emphasizing the need for thorough debugging and testing.

6. DINO: Due to computational constraints, the experiment reveals limitations in training the DINO model for the specified number of epochs. This underscores the necessity of considering available resources and time constraints when planning and executing experiments with computationally intensive models.

7. Self-Supervised Models: These experiments shed light on the challenges, limitations, and considerations involved in utilizing self-supervised learning models for the Pokemon classification task. Further investigation and optimization are recommended to maximize the performance and efficiency of these models in real-world applications.self-supervised learning models offer the potential advantage of learning representations directly from unlabeled data, eliminating the need for large labeled datasets. They can capture meaningful features and patterns that may be more specific to the task at hand. However, the experiments with SimCLR, SimSiam, BYOL, and DINO highlight the challenges, including long

training times, fluctuations in accuracy, and architectural limitations, which require careful tuning and optimization.

8. Advantages of the VGG as Transfer learning: Transfer Learning with VGG offers several advantages over self-supervised learning models for the Pokemon classification task. Firstly, it significantly reduces training time as the pre-trained VGG model already captures relevant features, enabling faster convergence. Secondly, transfer learning with VGG achieves relatively high accuracy within a shorter training period, as evidenced by reaching 48% accuracy within 20 epochs and approximately 70% accuracy after 200 epochs.

9. Disadvantages of the VGG as Transfer learning: Transfer learning with VGG also has some limitations. One major drawback is its dependence on pre-existing labeled datasets, which may not cover all the diverse classes and variations encountered in the Pokemon dataset. This can lead to suboptimal performance on unseen or challenging samples. Additionally, VGG's architecture may not be specifically tailored for the Pokemon classification task, resulting in limited flexibility and adaptability to the dataset's unique characteristics.

References

[1] Abhronil Sengupta, Yuting Ye, Robert Wang, Chiao Liu, and Kaushik Roy. Going deeper in spiking neural networks: Vgg and residual architectures. *Frontiers in neuroscience*, 13:95, 2019.

[2] Khoi Nguyen, Yen Nguyen, and Bao Le. Semi-supervising learning, transfer learning, and knowledge distillation with simclr. *arXiv preprint arXiv:2108.00587*, 2021.

[3] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 15750–15758, 2021.

[4] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33:21271–21284, 2020.

[5] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9650–9660, 2021.