## Data Preprocessing

### Data Overview

There are 4xxx images among 8 classes in our dataset. We split the dataset into two subsets: the training set accounts for 80% of the dataset, and the remaining is the validation set. These numbers follow the best practices in training machine learning model [].

### EDA

### Data Generation

Figure 1 indicates there are two problems in our dataset. First, xxx is the class having the highest number of samples (xxx), while xxx has the least figure (xxx). Therefore, our dataset is not general enough because according to this resource, 1000 images per class is suggested for training a deep model. Secondly, our dataset is imbalanced because there is a huge difference between these two above classes. We need to address this problem because the low generalization of the dataset can cause underperformance of our model due to biases. To overcome this situation, our target is to generate about 1500 images for each type of flower. First, we use the Selenium package to scrap the data from popular flower shops. This is the most effective way to gain more images, enriching the generalization of the trained model. We estimate that we collected more than 200 images in each class; these numbers are not enough to fulfill our dataset. Hence, we apply image augmentation on all existing images. Next, we augment these images by using the Albumentations package. This method increases the number of samples and diversify the labeled training sets by transforming the input image into vary forms []. After applying image augmentation, we have 11xxx images in our dataset. Figure 2 illustrates the distribution of image in each class after generation, and the number of images in each class ranges from xxx to xxx. Moreover, another advantage of image augmentation is that we create some images under a special condition, such as sun flare or low light. Therefore, adding these images can improve our models’ generalization because these models were trained under a more complicated data. Improving generalization is important because it helps model improve its performance in real life.

## Task 1: Classify Flower Images

### Evaluation Framework

There are three reasons why we choose accuracy as our main metric to evaluate model performance. The first reason is that the problem of imbalanced dataset has been resolved in the previous section. Hence, the situation of a model achieving a high accuracy by overfitting a specific class is unlikely. Moreover, we do not consider the importance of a particular kind of flower. Therefore, the sake of recall or precision is not as important as accuracy. If we are approaching a medical based problem, where the crucial of a specific class is highly recognized, and the dataset is imbalanced, recall or precision must have a higher priority than accuracy. Finally, we found that accuracy is the main metric to evaluate the Flower-102 dataset, which is an official dataset for evaluating a novel architecture.

### Strategy

We follow Karpathy’s recipe for training a model: baseline models, overfit, and regularize. Therefore, we overfit a particular model on the dataset until it achieves the train accuracy of more than 90%. For implementing this method, the model must learn on an uncomplicated dataset. Hence, we train the model on the generated dataset without applying any augmentations. This way helps the model gain a high accuracy faster compared to applying any image augmentation methods. Moreover, the model has learned some characteristics of the dataset before. Therefore, it can converge more quickly during learning some harder datasets (by applying image augmentation). This idea is like that of domain adaptation because the model has gained some knowledge about data distributions. After that, we gradually increase the strength of the augmentation. This helps the model performance gain the complex features of the dataset optimally and avoid the situation of overfitting during training.

### Configurations

#### Batch Normalization (BN)

Internal covariate shift is a phenomenon in which the distribution of input of each layer changes during training a deep Neural Network [1]. This will slow down and make it harder to train models. One way to resolve this is to normalize layers input and to say in other words, we normalize a part of the structure of the models and it is called Batch Normalization [2]. It acts as a regularizer and helps models to achieve the same accuracy with fewer training steps, hence, saving time as well [3].

#### Dropout

In the process of training a Neural Network, when the data is limited, our model can be overfitted, which results in bad performances. There have been many methods to reduce or to prevent overfitting in the model such as weight penalties L1 and L2 regularization and soft weight sharing [4]. However, to save time finding optimal hyperparameters for each different structure to regularize, dropout technique is introduced to address this problem. This technique includes dropping out a unit, temporarily removing it out of the network, along with its connections as well, refer to Fig. 1. However, there are drawbacks when applying dropout. It will increase your training time so in other words, applying dropout also means creating a trade-off between accuracy and training time.

#### AdamW and SGD

AdamW stands for Adam with decoupled weight decay. Both L2 regularization and weight decay regularization aims to reduce overfitting in models but how they work are slightly different. In a conference paper at ICLR 2019 by Loshchilov & Hutter, Adam with L2 regularization had been proved to have worse generalization performance than AdamW [6]. The paper also mentioned that AdamW had yielded better training loss and test error than Adam [7].

#### GeLU

GELUs stands for Gaussian Error Linear Units and it is a high-performing neural

network activation function [8]. As the Neural Network goes deeper and deeper while training, sigmoid activation has been proved to be less effective than RELUs (Rectified Linear Units), one of the most popular Machine Learning activation [9]. Based on the success of RELUs, ELUs was introduced as an activation to increase training speed. Based on the paper of Hendrycks & Gimpel in 2020, GELUs had been proven to yield higher accuracy than RELUs and ELUs.

#### He Initialization

He initialization, also known as Kaiming initialization is an initialization introduce in 2015 [10]. This method will allow us to train deep rectified models from scratch and look into deeper architectures [11]

#### Cross Entropy Loss

### Architectures

#### AlexNet

AlexNet, the first ILSVRC winner, is a groundbreaking convolutional neural network architecture that changed the field of deep learning. It introduced the ReLU activation function, which accelerates network speed by enhancing gradient flow. AlexNet offers several advantages, including direct image input into its classification model and the utilization of well-known techniques like dropout, GPU processing, parallelization, and ReLU. Furthermore, convolution layers can extract picture edges automatically, and fully-connected layers can learn these features. However, compared to newer models like ResNet, GoogLeNet, and VGGNet, AlexNet has a shallower architecture, and the use of larger convolution filters (5x5) has been discouraged.

#### VGG

VGG is a multi-layered deep CNN architecture that utilizes small convolution kernels to boost network depth. VGG has advantages over AlexNet since its small kernel size reduces the number of parameters and complexity significantly, resulting in faster learning. Moreover, VGG incorporates convolution layers to enhance non-linearity features and make reliable decisions. VGG-11, with its 11 weighted layers, including convolutional and fully connected layers, allows for learning more complex features. However, drawbacks include longer training time and high computational cost due to the model's depth and fully-connected nodes.

#### MobileNetV2

MobileNetV2 is a lightweight deep neural network architecture that utilizes depth-wise separable convolutions to reduce inference time compared to traditional convolutions. This approach replaces the traditional convolution with two independent operations: depth-wise convolution and pointwise convolution. The depth-wise convolution applies an identical filter to each input channel, while the pointwise convolution combines the outputs with a 1×1 convolution. This technique reduces parameters while maintaining accuracy. Moreover, an improvement of MobileNetV2 compared to the previous one is it applied the Inverted Residual, which is a short-connection technqiue has been introduced in the ResNet architecture. This technique helps prevent gradient vanishing, ensuring the loss value does not saturate after a few epochs. However, MobileNet-based architecture involves an trade-off between accuracy and inference time.

#### EfficientNetV2

EfficientNetV2 is an advanced CNN architecture that improves upon its predecessor, EfficientNet. It introduces "EfficientNetV2 Compound Scaling," which scales the network's depth, width, and resolution simultaneously, ensuring optimal performance across different computational resources. The EfficientNetV2 architecture is divided into multiple stages, each containing a series of convolutional layers, non-linear activation functions, and down-sampling operations. These stages are organized in a hierarchical manner, capturing both low-level and high-level visual features for improved discriminative power. Advantages of EfficientNetV2 include its efficient design philosophy, achieving excellent performance with fewer parameters, and the integration of Residual Channel Attention Network (ReCAN), enhancing accuracy by focusing on informative regions while suppressing noise or irrelevant features. However, architecture has drawbacks. Training and inference times are longer due to increased depth and width. The complex architecture may be difficult to interpret. Furthermore, performance relies on the quality and diversity of the training data, limiting performance on dissimilar datasets and emphasizing the importance of data representation and transfer learning.

#### DarkNet53

DarkNet53 is a convolutional neural network architecture initially introduced as the foundation of YOLOv3 in 2016. It builds upon DarkNet19 from YOLOv2. DarkNet53 gets its name from its 53 convolutional layers. Its design philosophy emphasizes simplicity and efficiency by stacking multiple layers with small filter sizes to capture complex features while minimizing computational complexity. Advantages of DarkNet53 include the inclusion of residual connections that address the vanishing gradient problem and facilitate training of deeper networks. It also employs feature fusion from multiple scales, enabling detection of objects of varying sizes while preserving fine-grained details and global context. However, a limitation of DarkNet53 is that, despite using skip connections to capture contextual information, it may not capture long-range dependencies as effectively as some other architectures.

#### CSPDarkNet53

CSPDarkNet53 is an extension of the DarkNet architecture that implements Cross Stage Partial Network (CSPNet) across different stages of the network. These connections allow the flow of information between early and late stages, facilitating better information propagation and feature reuse. The CSPDarkNet53 architecture employs a CSPNet strategy to partition the feature map of the base layer into two parts and then merges them through a cross-stage hierarchy. CSPDarkNet53 provides a training time advantage, allowing for faster convergence and reduced training durations compared to other architectures. However, a drawback of CSPDarkNet53 is the trade-off in accuracy, as it may not match the levels achieved by some alternative architectures, limiting its applicability.

### Ultimate Judgement

### Evaluation

## Task 2: Recommend Flower Images

### Evaluation Framework

It is prevalent for using precision and recall rates to measure the performance of a Content-based Image Retrieval (CBIR) system. Precision (specificity) measures the capability of the model to retrieve images like the given input image. Regarding recall, it assesses the ability to retrieve related images compared to the whole related images in the saved database. The reason why we do not consider accuracy is because accuracy makes the CBIR system become a binary classifier with two classes (relevant and non-relevant). Moreover, accuracy can lead to misunderstanding by recognizing all items as non-relevant because the number of non-relevant samples is significantly larger than the figure of relevant ones. Furthermore, this is not an effective way in image retrieval since the user always wants to acquire the relevant information.

There are some popular metrics for image comparison, such as L1, L2, Cosine, and Coefficient. The problem of using L1 and L2 for measuring image similarity is that it cannot handle the problem of brightness variance in the images. Moving onto Cosine, although this metric can solve the problem of bright contrast in image comparison, but it is sensitive when there are biases in the brightness of such images Regarding Cosine and Coefficient, although Cosine Similarity can solve the problem of bright contrast, we choose Coefficient as the main metric because this metric can handle the problem of bias effectively. This advantage of Coefficient makes it the best metric for solving the problem of brightness variance in assessing image similarity.

### Architecture

The issue of these above metrics is they cannot effectively indicate the images’ features. This is because these metrics evaluate the image similarity based on pixel-wise-pixel. Moreover, these pixels are nearly independent of each other, which makes it hard for such metrics to indicate the characteristics of each image. Another problem of applying those metrics naively is high computational resources. The reason is these metrics evaluate all the pixel of the image, leading to wasting computational resources. Hence, we resolve the problem of extreme calculation by implementing an encoder to reduce the image dimensionality. The encoder reduces the dimension of each image. Thus, our metrics can perform their calculation on a smaller amount of data, declining the resources for the evaluation process. Additionally, the information after reducing is more correlated with each other. This means that this information captures the characteristics of each image sharply. At this moment, our metrics can evaluate the similarity more precisely.

A picture containing text, screenshot, font, rectangle

Description automatically generated

Figure 1 CBIR System Architecture

Figure X shows our CBIR architecture. To begin with, we apply an extractor to our image database. At this stage, the extractor extracts the feature’s characteristics, indexes all of images, and stores the extracted information into a local file. The reason we precaculate the image’s features is we want to significantly avoid the number of redundant calculation resources for each image query. Next, for each time retrieving images, the extractor gains the input image’s characteristics before calculating the similarity score between each image. Next, we apply the K-nearest neighbor algorithm to find the 10-most similar images, rank all of them, before returning the results. Below we represent two ways we implement the extractor.

#### Extractor as Convolutional Layers in CNN Classifier

We implement these layers in task 1 for reducing the image dimensionality to indicate the features of each image. This is because these networks have learnt the dataset’s features by performing the classification task. Therefore, they have some understanding about the distribution of our dataset and make suggestions based on these insights. The advantage of this method is that we can utilize the models in task 1 for completing the mission of task 2.

#### Extractor as Encoder in Autoencoder

High-dimensional data can be converted to low-dimensional codes by training in a multilayer network with a small central layer to reconstruct high-dimensional input vectors. Gradient descent can be used for fine-tuning the weights in such “autoencoder” networks, but this works well only if the initial weights are close to a good solution. We describe an effective way of initializing the weights that allows deep autoencoder networks to learn low-dimensional codes that work much better than principal components analysis as a tool to reduce dimensionality of data.

Dimensionality reduction facilities the classification, visualization, communication, and storage of high-dimensional data. A simple and widely used method is PCA, which finds the directions of greatest variance in the dataset and represents each data point by its coordinates along each of these directions. We describe a nonlinear generalization of PCA that uses an adaptive, multilayer encoder network to transform the high-dimensional data into low-dimensional code and similar “decoder” network to recover data form the code.

Starting with random weights in the two networks, they can be trained together by minimizing the discrepancy between the original data and its reconstruction. The required gradients are easily obtained by using the chain rule to backpropagate erro derivatives first through the decoder network and then through encoder network. The whole system is called an autoencoder and

For training the Autoencoder model, we use triplet loss to optimize the model performance. The traditonal logistic loss aims to identify the relationships between the given image and the similar images optimally. However, it cannot utilize the relationship between three instances: examplar, relevant instance, and non-relevant instance. Therefore, triplet loss can learn effectively the complex features between these three instances by maximizing the similarity scores between examplar-positive pairs and minimizing those figures between exemplar-negative pairs. Therefore, the network can suggest the similar images more actively.

### Ultimate Judgement

### Evaluation

**References**

[1] [2] [3] S. Ioffe and C. Szegedy, “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”, Mar. 2015. doi: 10.48550/arXiv.1502.03167.

[4] [5] N. Srivastava, G. Hinton, A. Krizhevsky, L. Sutskever, and R. Salakhutdinov, “Dropout: a simple way to prevent neural networks from overfitting”, vol 15, no. 1, pp. 1929-1958, 2014. [Online]. Available: <https://www.jmlr.org/papers/volume15/srivastava14a/srivastava14a.pdf?utm_content=buffer79b43&utm_medium=social&utm_source=twitter.com&utm_campaign=buffer>

[6] [7] I. Loshchilova and F. Hutter, “Decoupled weight decay regularization”, 2017, doi: 10.48550/arXiv.1711.05101

[8] [9] D. Hendrycks and K. Gimpel, “Gaussian Error Linear Units (GELUs)”, 2016, doi: 10.48550/arXiv.1606.08415

[10] [11] K. He, X. Zhang, S. Ren, and J. Sun, “Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification”, 2015, doi: 10.48550/arXiv.1502.01852