

|  |
| --- |
| AI Generated ||  Real |
|  |
|  |
|  |

* **Agenda**

1. **Introduction**
   * **Overview of the project**
   * **Objectives and goals**
2. **Data Description**
   * **Names of datasets used**
   * **Description of datasets**
   * **Data preprocessing steps**
3. **Model Training Process**
   * **Data loading and transformations**
   * **Neural network architecture**
   * **Training loop and optimization**
4. **Model Evaluation**
   * **Evaluation metrics**
   * **Confusion matrix**
   * **Classification report**
5. **Visualization of Results**
   * **Confusion matrix heatmap**
   * **Interpretation of results**

* **Introduction**

**AI-based image generation has continued to rapidly improve,**

**producing increasingly more realistic images with fewer obvious visual**

**flaws.**

**AI-generated images are being used to create fake online profiles**

**which in turn are being used for spam, fraud, and disinformation**

**campaigns.**

**As the general problem of detecting any type of manipulated or**

**synthesized content is receiving increasing attention, here we focus on**

**amore narrow task of distinguishing a real face from an AI-generated**

**face.**

* **Data Description:**

**In this project, various datasets consisting of art images were utilized. Each dataset includes images categorized into different classes. The classes represent distinct Faces, depending on the specific dataset. Data preprocessing involved resizing the images to a consistent dimension and normalizing the pixel values to ensure uniformity. The dataset was then split into training and testing subsets to facilitate effective model evaluation, datasets are :**

1. **140k Real and Fake Faces**

**This dataset consists of all 70k REAL faces from the Flickr dataset.**

**collected by Nvidia, as well as 70k fake faces sampled from the 1 million.**

**FAKE faces (generated by StyleGAN) that were provided by Bojan.**

**In this dataset, I convenient combined both datasets, resized all the**

**images into 256px, and split the data into train, validation, and test set. I**

**also included some CSV files for convenience.**

1. **CelebA-HQ resized (256x256)**

**A dataset containing 30,000 high-quality celebrity faces, resampled to 256px.**

**This dataset was used by NVIDIA in the research paper Progressive Growing of GANs for Improved Quality, Stability, and Variation.**

**It is meant for generative models. As such, it only has images and no attributes/labels.**

1. **Synthetic Faces High Quality (SFHQ) part 2**

**This dataset consists of 91,361 high quality 1024x1024 curated face images and was created by "bringing to life" various 3D models and correcting bad "text to image" generations from stable diffusion model using a process like what is described in this short twitter thread which involve encoding the images into StyleGAN2 latent space and performing a small manipulation that turns each image into a photo-realistic image.**

**The dataset also contains facial landmarks (extended set) and face parsing semantic segmentation maps. An example script is provided and demonstrates how to access landmarks, segmentation maps, and textually search withing the dataset (with CLIP image/text feature vectors) and performs some exploratory analysis of the dataset. link to GitHub repo of the dataset.**

1. **Face Dataset Using Stable Diffusion v.1.4**

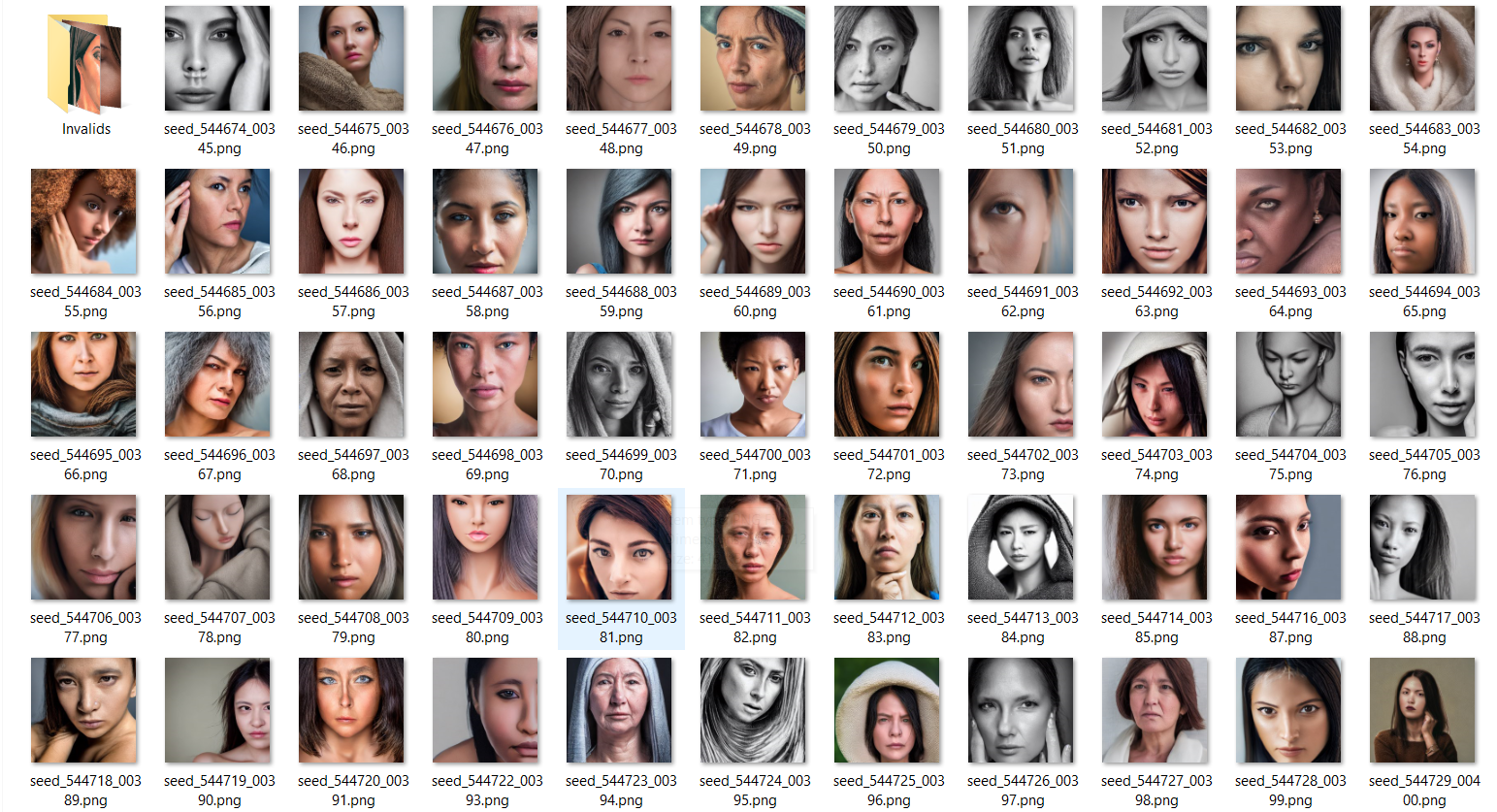
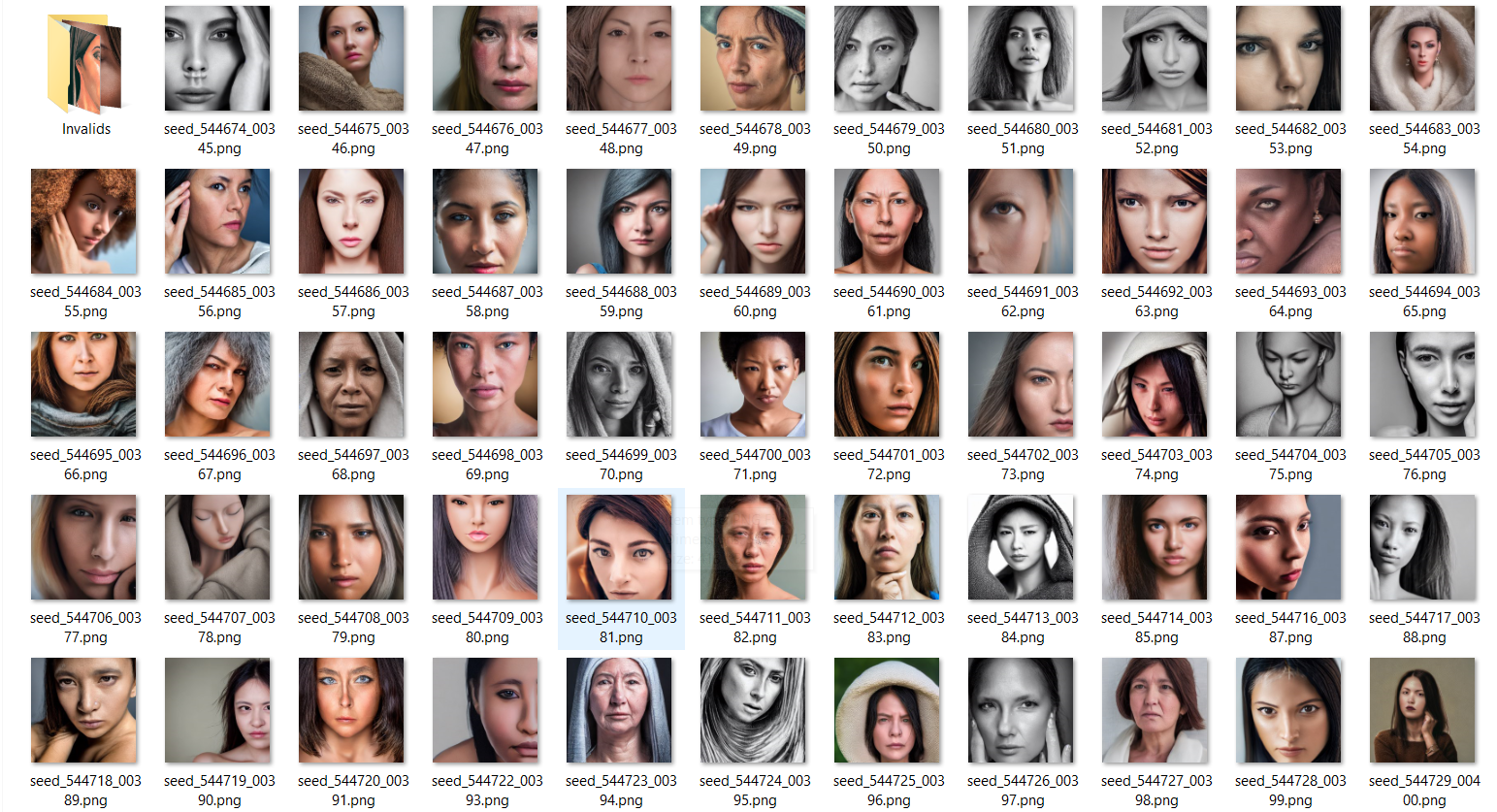
**This dataset consists of all 70k REAL faces from the Flickr dataset.**

**collected by Nvidia, as well as 70k fake faces sampled from the 1 million.**

**FAKE faces (generated by StyleGAN) that were provided by Bojan.**

**In this dataset, there are 2 combined datasets, resized all the images.**

**into 256px, and split the data into train, validation, and test set.**

****

1. **Stable Diffusion Face Dataset**

**Ai generated Human faces using Stable Diffusion 1.5, 2.1, and SDXL 1.0 checkpoint.**

**The main objective was to generate photos that were as realistic as possible, without any specific style, focusing mainly on the face.**

**Fake Ai generated Human faces.**

* **Images in 512x512px resolution were generated using SD 1.5;**
* **Images in 768x768px resolution were generated using SD 2.1;**
* **Images in 1024x1024px resolution were generated using SD XL 1.0;**

1. **Synthetic Faces High Quality (SFHQ) part 3**

**This dataset consists of 118,358 high quality 1024x1024 curated face images and was generated by StyleGAN2 generator with a new truncation trick to increase diversity, and a semi-manually curation post generation process to make sure all images are of high quality.**

**The new truncation trick utilizes the encoder4editing network that allows to correct StyleGAN2.**

**StyleGAN2 latent space using an encoder and performing a small manipulation that turns each image into a photo-realistic image.**

1. **Synthetic human faces for 3D reconstruction**

**The dataset is generated by drawing samples from the EG3D model, trained**

**initially on the FFHQ dataset. As a result, the synthetic face images of size**

**512x512 are high-quality realistic human faces.**

* **Model Training Process**

**-Data Loading and Transformations:**

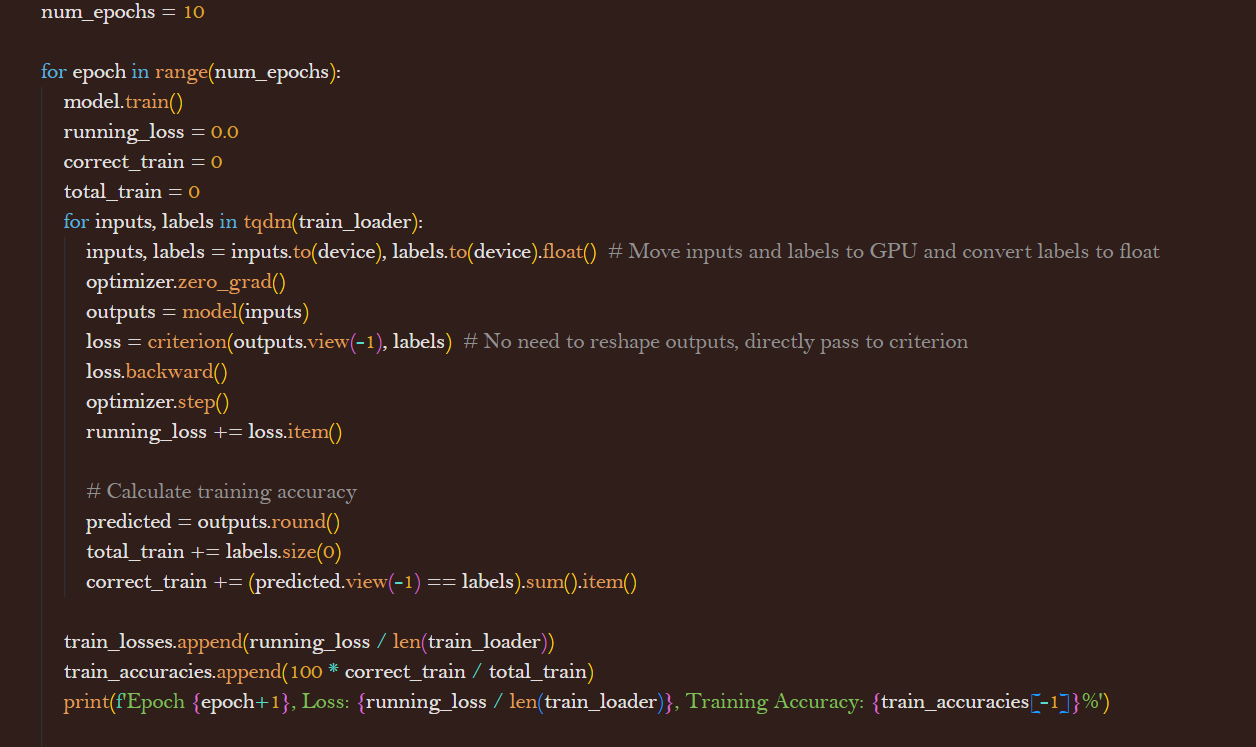
**The first step in the training process involves loading the datasets and applying necessary transformations. This includes resizing the images and normalizing pixel values. The dataset is divided into training and testing sets, enabling the evaluation of the model's performance.**

**-Neural Network Architecture:**

**A convolutional neural network (CNN) was employed to learn patterns in the images. The network architecture includes several convolutional layers with activation functions and pooling layers. These layers extract features from the images and reduce dimensionality, leading to a fully connected layer that performs the final classification**.

**- Training Loop and Optimization:**

**The training process involves multiple epochs where the model learns from the training data. Each epoch includes forward propagation for making predictions, loss calculation to measure the difference between predictions and actual values, and backward propagation for updating the model's weights. This iterative process continues until the model achieves satisfactory performance.**

****

* **Model Overview**

**The convolutional neural network (CNN) designed for the AI Art Graduation Project plays a central role in classifying art images. This section provides a detailed explanation of the model's architecture, key components, and its function in learning from the data.**

**Model Architecture**

**The architecture of the CNN consists of several layers, each serving a specific purpose in the feature extraction and classification process. The primary components of the model include convolutional layers, activation functions, pooling layers, and fully connected layers.**

**1. Convolutional Layers:**

**The model starts with convolutional layers that apply a set of filters to the input images. These filters move across the image, performing convolution operations to detect various features such as edges, textures, and patterns. Each convolutional layer is followed by an activation function to introduce non-linearity, allowing the model to learn more complex representations.**

**2. Activation Functions:**

**ReLU (Rectified Linear Unit) is used as the activation function after each convolutional layer. ReLU helps in speeding up the training process by allowing positive values to pass through while setting negative values to zero. This non-linearity is crucial for the model to learn intricate patterns in the data.**

**3. Pooling Layers:**

**Pooling layers, specifically max pooling, are used after certain convolutional layers to reduce the spatial dimensions of the feature maps. This downsampling helps in reducing the computational load and the number of parameters, thereby preventing overfitting. Max pooling captures the most prominent features by selecting the maximum value from each region of the feature map.**

**4. Fully Connected Layers:**

**After several convolutional and pooling layers, the model transitions to fully connected layers. These layers take the flattened output from the convolutional layers and pass it through a series of neurons. The fully connected layers act as the decision-making part of the network, combining the extracted features to classify the input images into the respective classes.**

**5. Output Layer:**

**The final layer of the model is a fully connected layer with a single neuron (for binary classification) or multiple neurons (for multi-class classification), followed by a sigmoid or softmax activation function, respectively. This layer outputs the probability distribution over the classes, indicating the model's confidence in each class prediction.**

**Key Features and Optimization**

**1. Weight Initialization:**

**Weights in the network are initialized using techniques like Xavier or He initialization to ensure that they start with appropriate values. Proper initialization helps in faster convergence during training.**

**2. Loss Function:**

**The loss function used in the model is Binary Cross-Entropy for binary classification or Categorical Cross-Entropy for multi-class classification. The loss function measures the difference between the predicted and actual labels, guiding the optimization process.**

**3. Optimizer:**

**An optimizer like Adam or SGD (Stochastic Gradient Descent) is used to update the weights of the network during training. The optimizer adjusts the weights based on the gradients calculated from the loss function, enabling the model to learn from the data iteratively.**

**4. Regularization:**

**Techniques such as dropout are employed to prevent overfitting. Dropout randomly sets a fraction of the neurons to zero during training, forcing the network to learn redundant representations and improving its generalization capability.**

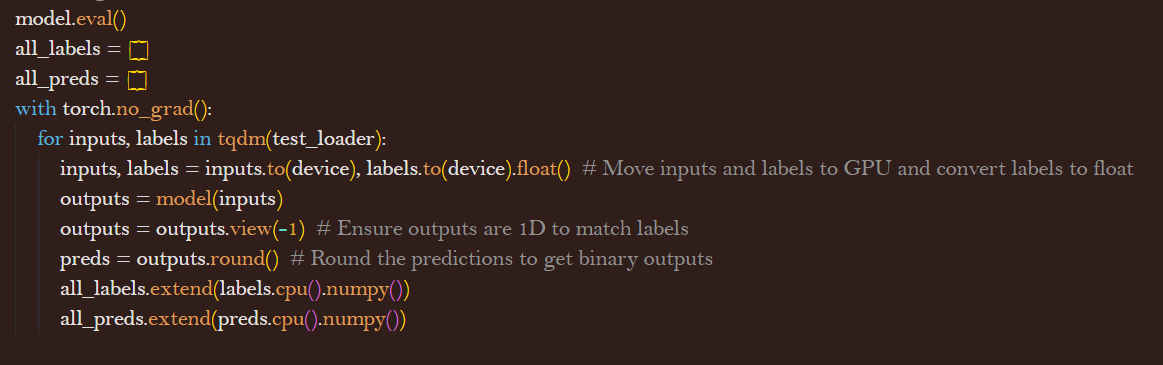
**5. Training Process:**

**The training process involves multiple epochs, where the model iterates over the entire training dataset. During each epoch, the model performs forward propagation to make predictions, computes the loss, and uses backward propagation to update the weights based on the gradients. This iterative process continues until the model achieves satisfactory performance on the training data.**

* **Model Evaluation**

**Evaluation Metrics:**

**To assess the model's performance, several metrics are used. These include accuracy, precision, recall, and F1-score. These metrics provide insights into the model's ability to correctly classify the images.**

****

#### Evaluation Metrics

**1. Accuracy:**

Accuracy is the ratio of correctly predicted instances to the total instances. It provides a general measure of the model's performance but might not be sufficient in cases of imbalanced datasets.

**2. Precision:**

Precision is the ratio of true positive predictions to the sum of true positive and false positive predictions. It measures the accuracy of the positive predictions, indicating how many of the predicted positive instances are positive.

**3. Recall:**

Recall, also known as sensitivity or true positive rate, is the ratio of true positive predictions to the sum of true positive and false negative predictions. It measures the model's ability to identify all positive instances in the dataset.

**4. F1-Score:**

The F1-Score is the harmonic mean of precision and recall. It provides a balance between the two metrics, A high F1-Score indicates that the model has both high precision and high recall.

* **Confusion Matrix**

**-The confusion matrix is a table that summarizes the performance of the model by displaying the counts of true positive (TP), true negative (TN), false positive (FP), and false negative (FN) predictions. It provides a detailed breakdown of the model's classification performance, highlighting areas where it performs well and where it struggles.**

* **True Positives (TP): Instances correctly predicted as positive.**
* **True Negatives (TN): Instances correctly predicted as negative.**
* **False Positives (FP): Instances incorrectly predicted as positive.**
* **False Negatives (FN): Instances incorrectly predicted as negative.**

**Visualization of Results**

**1. Confusion Matrix Heatmap:**

**A confusion matrix heatmap visualizes the confusion matrix, making it easier to interpret. The heatmap uses color coding to represent the counts of TP, TN, FP, and FN, highlighting the areas where the model performs well and areas that need improvement. This visual representation helps in quickly identifying the strengths and weaknesses of the model.**

**2. Precision-Recall Curve:**

**The precision-recall curve plots precision versus recall for different threshold values. It provides insights into the trade-off between precision and recall, helping in selecting the optimal threshold for making predictions.**

**3. ROC Curve:**

**The Receiver Operating Characteristic (ROC) curve plots the true positive rate (recall) against the false positive rate for different threshold values. The area under the ROC curve (AUC-ROC) indicates the model's ability to distinguish between positive and negative instances. A higher AUC-ROC value signifies better performance.**

**The confusion matrix heatmap highlights specific areas where the model might be struggling. For instance, a high number of false positives in a particular class indicates that the model is over-predicting that class. This information is crucial for diagnosing and addressing specific issues in the model.**

**Code:**

**A computer screen shot of a program

Description automatically generated**

**Visualization :**

**A blue squares with white text

Description automatically generated**

* **Classification Report:**

**The classification report includes detailed metrics such as precision, recall, and F1-score for each class. Precision measures the accuracy of positive predictions, recall evaluates the model's ability to identify all positive instances, and F1-score balances precision and recall.**

A black screen with white text

Description automatically generated

* **The PRE-Trained models**
* **EfficientNet-B2 convolutional neural network .**
* **Xception.**
* **VGG16**

**EfficientNet B2,** **VGG16 and Xception are powerful pre-trained models commonly used in deep learning projects for image classification. VGG16 is known for its simplicity and depth, making it effective but computationally intensive.**

**Xception, on the other hand, uses an innovative architecture with depth wise separable convolutions, offering high performance with fewer parameters. By leveraging these models through transfer learning, we can achieve high accuracy and efficiency in various image classification tasks, including art analysis in the AI Art Graduation Project.**

**And efficient model designed to achieve high performance with fewer computational resources. Its innovative use of compound scaling and MBConv blocks makes it an excellent choice for image classification tasks, including those involving complex and diverse datasets like art images. By leveraging EfficientNet B2 in transfer learning, the AI Art Graduation Project can benefit from the model's accuracy and efficiency, achieving robust results with optimized resource usage.**

1. **VGG16**

**Architecture:**

* **Convolutional Layers:** VGG16 uses 3x3 convolutional filters with stride 1 and padding to maintain the spatial resolution of the input. The network consists of 13 convolutional layers, grouped into 5 blocks, each followed by a max-pooling layer.
* **Max Pooling:** Each convolutional block is followed by a max-pooling layer with a 2x2 filter and stride 2, which reduces the spatial dimensions by half.
* **Fully Connected Layers:** After the convolutional blocks, the network includes three fully connected layers, the first two with 4096 neurons and the last one with 1000 neurons (for classification into 1000 classes).
* **Activation Function:** ReLU (Rectified Linear Unit) activation is applied after each convolutional and fully connected layer, introducing non-linearity to the model.
* **Softmax Layer:** The final layer uses the softmax activation function to produce a probability distribution over the classes.

**Usage and Adaptation:** In transfer learning, VGG16 can be used as a feature extractor by freezing the convolutional layers and replacing the fully connected layers with custom layers tailored to the specific task. This allows leveraging the rich feature representations learned from large datasets like ImageNet.

**Results:** Using VGG16 typically results in high accuracy for image classification tasks due to its deep architecture and extensive pre-training. However, it can be computationally expensive due to the large number of parameters.

1. **Xception**

**Overview:** Xception, which stands for "Extreme Inception," is a deep learning model proposed by François Chollet. It builds upon the Inception architecture but replaces the standard Inception modules with depthwise separable convolutions, making it more efficient and effective. Xception has demonstrated superior performance on various image classification benchmarks.

**Architecture:**

* **Depthwise Separable Convolutions:** Xception replaces traditional convolutions with depthwise separable convolutions, which split the convolution operation into two steps: depthwise convolution and pointwise convolution. This reduces the number of parameters and computational cost while maintaining performance.
* **Entry Flow:** The entry flow consists of a series of convolutional layers and max-pooling layers that gradually reduce the spatial dimensions and increase the depth of the feature maps.
* **Middle Flow:** The middle flow consists of multiple repeated blocks of depthwise separable convolutions. This part of the network captures high-level features and patterns.
* **Exit Flow:** The exit flow transitions from the middle flow to the final fully connected layers, further reducing the spatial dimensions and aggregating features for classification.
* **Activation Function:** ReLU is used as the activation function after each convolutional layer.
* **Global Average Pooling:** Instead of fully connected layers, Xception uses global average pooling to reduce the feature maps to a single vector per feature map, which is then passed to the final dense layer for classification.
* **Softmax Layer:** The final layer uses the softmax activation function to produce class probabilities.

**Usage and Adaptation:** Like VGG16, Xception can be adapted for specific tasks by freezing the pre-trained layers and adding custom layers on top. This allows the model to leverage its efficient feature extraction capabilities for various applications.

1. **EfficientNet-B2**

is a family of convolutional neural networks introduced by Mingxing Tan and Quoc V. Le in their paper "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks." EfficientNet B2 is one of the models in this family, known for balancing accuracy and efficiency through a novel compound scaling method.

**Architecture**

EfficientNet B2 is built upon three main scaling dimensions:

1. **Depth:** Number of layers in the network.
2. **Width:** Number of channels in each layer.
3. **Resolution:** Input image size.

The compound scaling method uniformly scales these dimensions to achieve better performance with fewer resources.

**Key Components:**

* **Mobile Inverted Bottleneck Convolution (MBConv):** EfficientNet uses MBConv blocks, which include depthwise separable convolutions and squeeze-and-excitation (SE) blocks. These components help in reducing computational complexity while maintaining high accuracy.
* **Swish Activation Function:** EfficientNet employs the Swish activation function (x \* sigmoid(x)), which improves model performance by providing smooth non-linearities.
* **Compound Scaling:** EfficientNet B2 scales the baseline EfficientNet architecture using a specific compound coefficient to balance the depth, width, and resolution.

**MBConv Block Structure:**

1. **Expansion Phase:** Expands the input channels using 1x1 convolutions.
2. **Depthwise Convolution:** Applies depthwise separable convolutions, which operate on each input channel separately.
3. **SE Block:** Squeeze-and-Excitation block recalibrates the feature maps by weighting them according to their importance.
4. **Projection Phase:** Reduces the channels back to the desired number using another 1x1 convolution.

**EfficientNet B2 Configuration:**

* **Input Resolution:** 260x260 pixels.
* **Depth:** Moderately increased number of layers compared to B0.
* **Width:** Increased number of channels in each layer.
* **Number of Parameters:** Approximately 9.2 million.

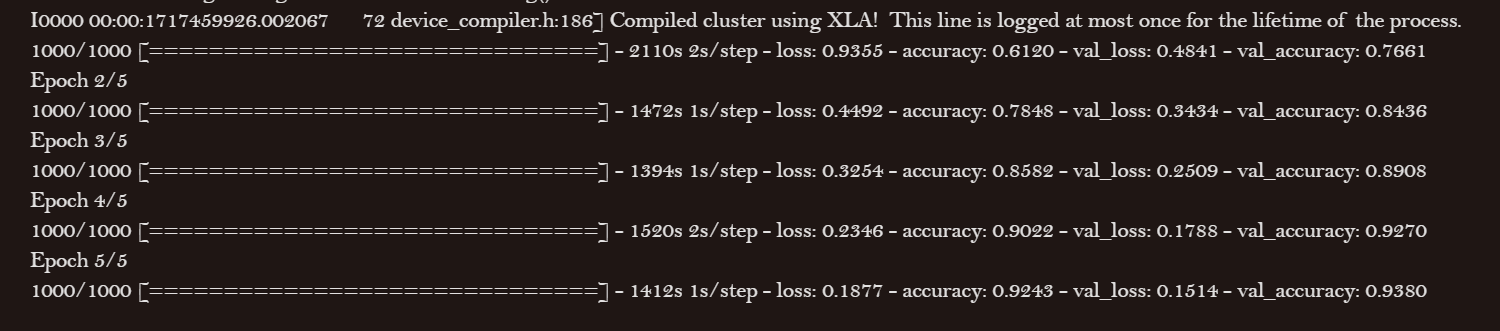
**Usage and Adaptation**

EfficientNet B2 can be used for transfer learning by leveraging the pre-trained weights from large datasets such as ImageNet. The typical process involves:

1. **Loading the Pre-trained Model:** EfficientNet B2 is loaded with pre-trained weights, which have learned to recognize a wide variety of features from a large and diverse dataset.
2. **Modifying the Output Layer:** The final layer, which is tailored to the number of classes in the pre-trained dataset, is replaced with a new layer corresponding to the specific number of classes in the target task.
3. **Freezing Initial Layers:** The initial layers are frozen to retain the learned features and prevent them from being updated during training.
4. **Training Custom Layers:** Only the newly added layers are trained on the target dataset, allowing the model to adapt to the specific task with fewer data and computational resources.

**Pre-Trained Models Results :**

1. **VGG16**
   * **Training**

****

* + **Test**

**A black background with white numbers

Description automatically generated**

1. **Xception :**
   * **Training:**

A black screen with white text

Description automatically generated

* + **Test**

A number on a black background

Description automatically generated