AMNIL TECHNOLOGIES

# Presentation on Image Classification using EfficientNet-B0 on Tiny ImageNet

Prepared by: Sejal Panta
28th Oct, 2025

## What is Image Classification?

Image Classification is the task of teaching a computer to look at a picture and tell what it is, essentially assigning a label or category to an image based on its visual content.

It involves classifying an entire image into one of several predefined categories and is one of the fundamental problems in computer vision, serving as a building block for many other vision tasks such as object detection and image segmentation.

Example: An image of a dog is correctly classified under the label “dog” from categories like cat, dog, horse, etc.

## Image Classification Architectures

1. Era 1 (2012-2014): Classical CNNs
   Example: AlexNet, VGGNet – simple deep convolutional networks.

2. Era 2 (2015-2017): Residual Learning
   Example: ResNet, DenseNet – skip connections to train very deep networks.

3. Era 3 (2017-2019): Efficient Architectures
   Example: MobileNet, ShuffleNet, EfficientNet – lightweight, efficient, suitable for mobile/edge devices.

4, Era 4 (2020-Present): Attention & Transformers
   Example: Vision Transformers (ViT), Swin Transformer, ConvNeXt – use self-attention for global feature learning.

## Era 3 – Efficient Architectures (2017-2019)

Goal: High accuracy with low computational cost.

Techniques:

- Depthwise separable convolutions (MobileNet)
- Channel shuffle (ShuffleNet)
- Compound scaling (EfficientNet)

Impact: Enables training and deployment on resource-constrained devices without sacrificing performance.

## EfficientNet (2019)

EfficientNet is a family of convolutional neural networks (CNNs) designed by Google AI in 2019 for image classification. It was created to achieve high accuracy while using much fewer parameters and less computation compared to older models like ResNet, Inception, or VGG.

Before EfficientNet, people tried to make models more powerful by:

- Making them deeper (adding more layers) e.g. ResNet
- Making them wider (adding more filters) e.g. Wide ResNet
- Using higher image resolution

But doing these separately often made models inefficient, wasting memory and computation without proportional accuracy gain.

EfficientNet solved this by introducing a balanced scaling method that increases depth, width, resolution in a uniform and efficient way.

Key Idea: Compound Scaling

EfficientNet uses a compound scaling formula:
depth=$\alpha^{\phi}$, width=$\beta^{\phi}$ , resolution=$\gamma^{\phi}$

where,
$\phi$ = scaling factor,
$\alpha$, $\beta$, $\gamma$ are constants found through experiments.

This means instead of randomly making the model deeper or wider, EfficientNet balances all three properly.

Example:
If you want to make a small model (B0) into a bigger one (B1–B7), EfficientNet scales it systematically using this formula.

**EfficientNet-B0**

It is the base architecture from which all other EfficientNets (B1–B7) are scaled.

Example:

- EfficientNet-B0: Base model (224×224 input)
- EfficientNet-B1: Scaled up (240×240 input, slightly deeper)
- EfficientNet-B7: Very large version (600×600 input, deeper + wider)

All use the same architecture pattern, i.e. they are only scaled versions of B0.

It was created using Neural Architecture Search (NAS), an automated AI method that finds the best network structure for accuracy vs. efficiency.

Key Principles in EfficientNet-B0:

1. MBConv Blocks:
   Light-weight blocks with depthwise and pointwise convolutions to reduce computation.

2. Swish Activation:
   Uses Swish(x) = x * sigmoid(x) instead of ReLU for smoother gradients and better accuracy.

3. Squeeze-and-Excitation (SE) Blocks:
   These help the network focus on important features in channels.

4. Batch Normalization after each convolution for stable training.

5. Compound Scaling:
   All other models (B1–B7) are created by scaling B0's depth, width, and resolution using the formula.

Why EfficientNet-B0 for Tiny ImageNet?

- Lightweight & fast: suitable for small datasets like Tiny ImageNet
- Pretrained on ImageNet: enables transfer learning
- Achieves high accuracy with fewer parameters (~5.3M)
- Ideal for Colab GPU / resource-limited setups

Project Pipeline (Tiny ImageNet):

1. Dataset Preparation: Tiny ImageNet (200 classes, 64×64 images)
2. Model Selection: EfficientNet-B0 pretrained on ImageNet
3. Transfer Learning: Replace final layer into 200 classes
4. Training: Fine-tune on Tiny ImageNet
5. Inference: Classify new images with predicted labels

Key Advantages:

- Efficient use of GPU memory and computation
- High accuracy with small models
- Scalable from B0 to B7 depending on resources
- Easy integration with PyTorch / TensorFlow

Key Disadvantages:

- Sensitive to input resolution and scaling
- Requires careful hyperparameter tuning
- Limited interpretability
- Pretrained models may not generalize perfectly to very different domains

## Wnid to class name mapping of different classes:

| wnid | Class Name |
|---|---|
| n01443537 | Goldfish |
| n01770393 | Scorpion |
| n01855672 | Goose |
| n01882714 | Koala |
| n02056570 | King Penguin |
| n02099601 | Golden Retriever |

# Thank you!