Lab 1 Report : Image Classification with CNNs in Tensorflow

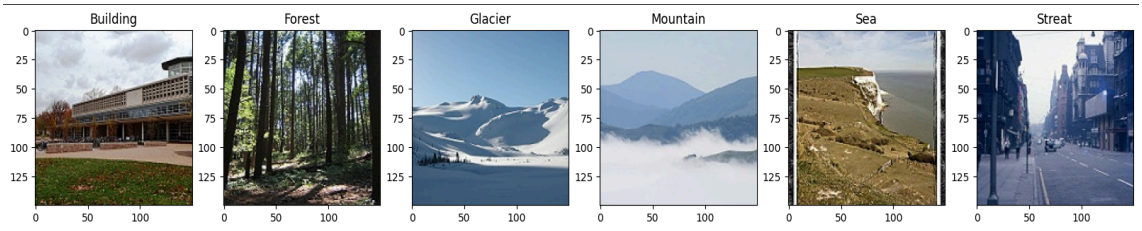
Nguyễn Triều Vương
Student ID: SE182185
FPT University - DAT301m

1. Introduction

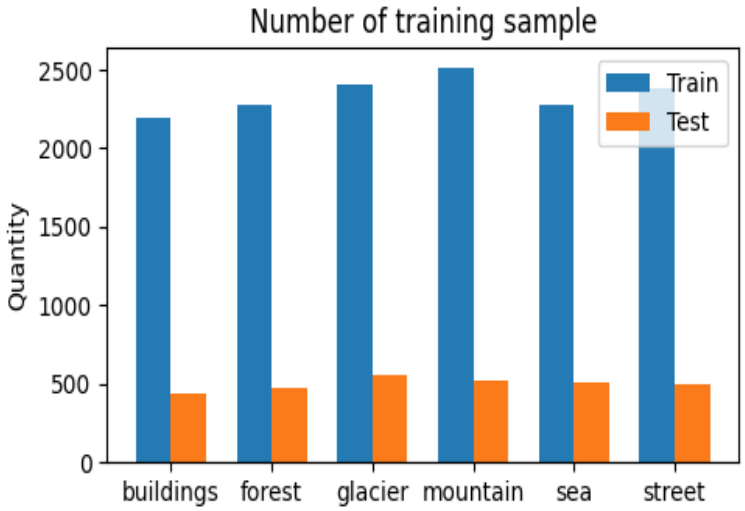
This report aims to analyze and visualize the efficiency of three CNN based models for a classification task to experience how to enhance the performance of a model using following provided techniques such as data preprocessing, data augmentation, hyperparameters tuning, and adding dropout, batch normalization, regularizations layers.

2. Dataset

Intel Image Classification is the dataset used in this lab, it included 25000 images of size 150x150 distributed under 6 categories: buildings, forest, glacier, mountain, sea and street. There are around 14000 images in Train, 3000 in Test and 7000 in Prediction.



Training and testing data distribution



3. Data Preprocessing

After loading the dataset using ImageDataGenerator to make it more consistent I resized them to 150x150 and normalised every image to a scale from 0 to 1.

4. Data Augmentation

To diversify the dataset, some data augmentation techniques were used such as :

+ Random flip (horizontal and vertical)
+ Random rotation
+ Random translation
+ Random Zoom

5. Fist Model

5.1 Convolutional and Pooling Layers

Model has four convolution blocks each consisting of:

+ A Conv2D layer with increasing filters (32, 64, 128, 128), kernel size (3, 3) and ReLU activation.

+ A Maxpooling2D layer with pool size (2, 2) to reduce spatial dimensions.

This structure helps in capturing spatial hierarchies of features and progressively reduces the spatial size of the feature maps.

5.2 Fully Connected Layers

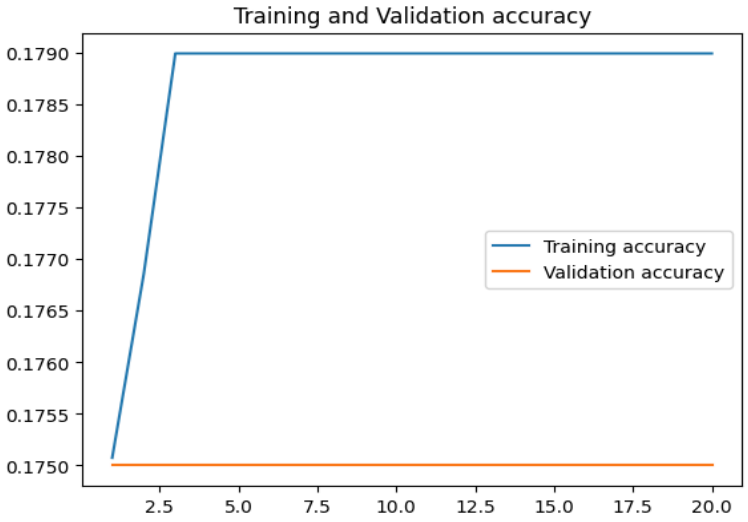
After flattening the output from convolutional layers:

+ Dense Layer 1: 512 units, ReLU activation
+ Dense Layer 2: 256 units, ReLU activation
+ Output Layer: 6 units, softmax activation - representing the 6 output classes

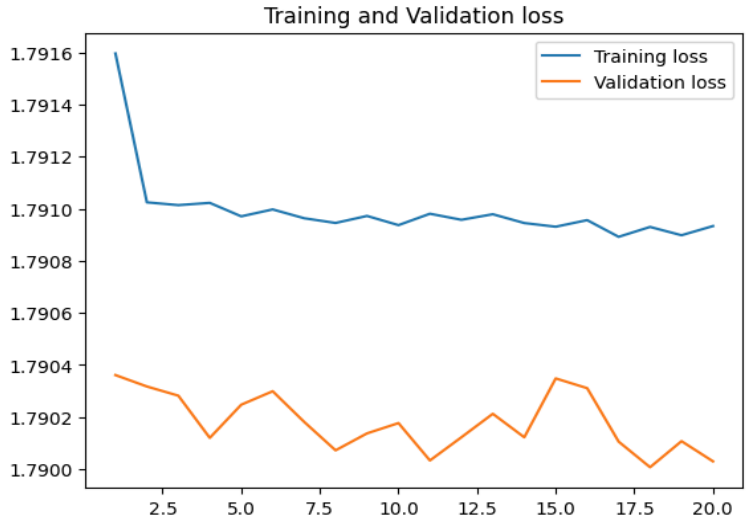
5.3 Training Details

- Loss function: Categorical Crossentropy
- Optimizer: Adam with 0.001 learning rate
- Metrics: Accuracy
- Epochs: 20
- Batch Size: 32

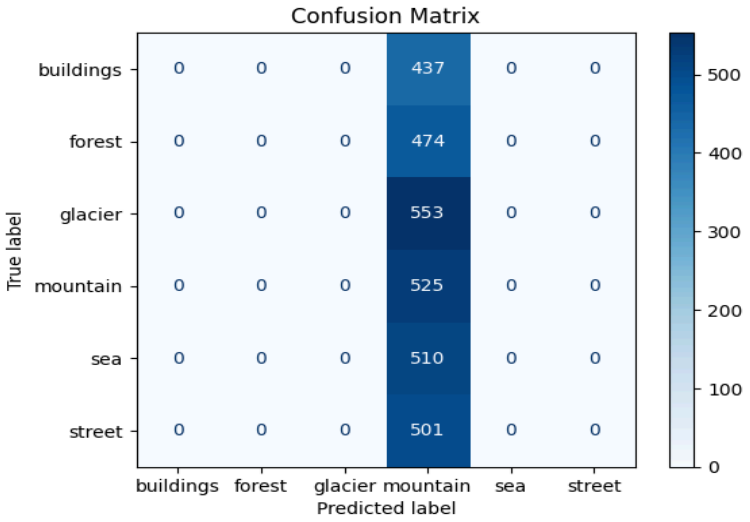
5.4 Results



Training accuracy quickly plateaus (~17.9%), and validation accuracy remains flat (~17.5%) — indicating no real learning.



Both losses remain almost constant, showing no significant improvement.



The model predicts only one class (mountain) for all inputs,meaning it's completely failing to differentiate classes.

6. Enhanced Model

6.1 Adding more layers

In order to enhance model's performance, following mechanism were used to create a better mode:

+ Padding same
+ Norm l2 Regularizer

Moreover, Batch Normalization layers were added to stabilize and accelerate the training process before ReLU activation and MaxPooling2D layer.

6.2 Convolution Block

Model included four convolution block each has:

+ Conv2D layer with filters, kernel size = 3, padding ='same', norm l2 regularizer
+ Batch Normalization layer
+ ReLU activation layer
+ MaxPooling2D layer

6.3 Fully Connected Block

After flattening the output from convolutional layers, it will go through two dense layers each with units, norm l2 regularizer, Batch Normalization, ReLU activation and Dropout layer.

Then it was passed to the output layer of 6 units and a Softmax activation function to classify the desirable output.

6.4 Training Details

- Loss function: Categorical Crossentropy
- Optimizer: Adam
- Metrics: Accuracy
- Epochs: 30
- Batch Size: 32

6.5 Hyperparameters Tuning

Keras tuner was the library used for tuning hyperparam, in this scenario following parameters tuned were:

- Conv2D [64, 64, 512, 512] for each layer in order
- Dense Units [1028, 1028] for each layer before the output
- Weight Decay for norm l2 regularizer : 0.001
- Learning Rate for optimizer : 0.0001
- Dropout Rate for each Dense layer : 0.3

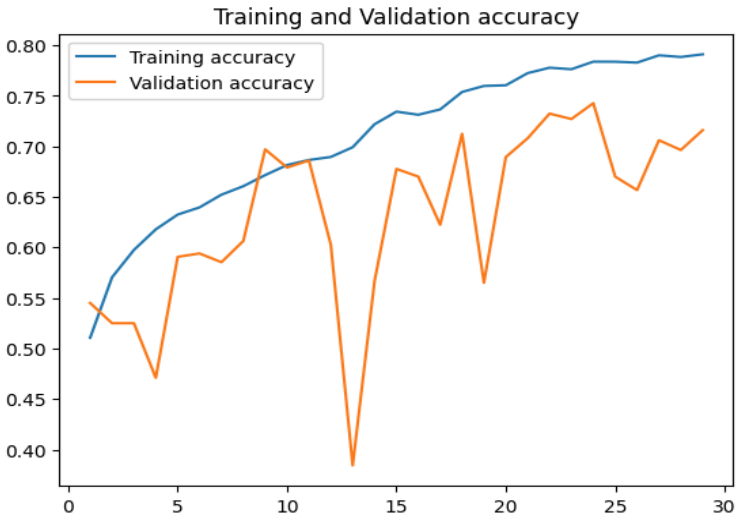
The tuning output shown that provided parameters are suitable to train

6.6 Callbacks

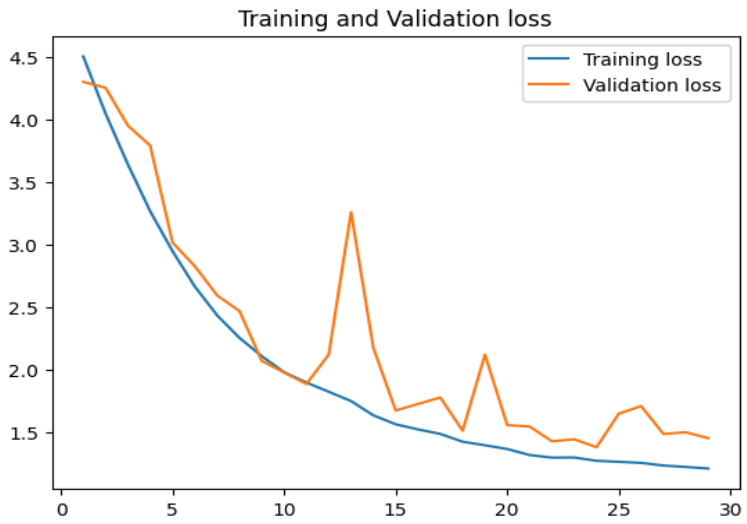
- ReduceLROnPlateau reduces the learning rate automatically when the model's performance stops improving.

- EarlyStopping stops training early if the model is not improving, to save time and prevent overfitting.

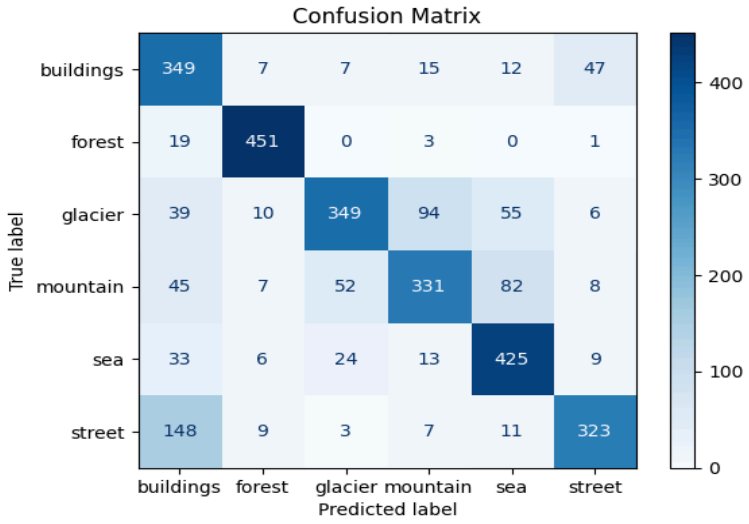
6.7 Results



Training accuracy steadily improves and reaches ~0.79. Validation accuracy is lower (~0.72) and fluctuates, suggesting some overfitting.



Training loss decreases consistently. Validation loss also drops but has some spikes, confirming overfitting signs.



The model performs well on forest and sea classes but often confuses streets with buildings, and glaciers with mountains.

Answer Question 1:

- Padding 'same' keeps the output size the same as the input size by adding zeros around the input so that the convolution does not shrink the spatial dimensions.

- L2 regularization reduces overfitting, promotes smaller weights, improves training stability, encourages simpler models, and enhances convergence during training.

- Batch normalization speeds up training, stabilizes gradients, reduces overfitting, allows higher learning rates, and makes the model less sensitive to weight initialization.

- Dropout is a regularization technique that randomly deactivates neurons during training to prevent overfitting, improve generalization, and make the network less reliant on specific neurons.

- Overall, each block in the enhanced architecture is the same with the first model. However, adding padding, regularizer, batch normalization and dropout to the model helps it converge better during the training process with a higher accuracy and lower loss after tuning to search for suitable hyperparameters.

7. EfficientNet

7.1 Introduction

For further improvement, EfficientNet-B0 replicated a high-performance and resource-efficient CNN architecture. The goal is to reproduce its structure and evaluate its effectiveness in image classification tasks.

EfficientNet-B0 Architecture included:

1. Conv3x3, 150x150 resolution, 32 channels, 1 layer
2. MBConv1, kernel 3x3, 112x112 resolution, 16 channels, 1 layer
3. MBConv6, kernel 3x3, 112x112 resolution, 24 channels, 2 layers
4. MBConv6, kernel 5x5, 56x56 resolution, 40 channels, 2 layers
5. MBConv6, kernel 3x3, 28x28 resolution, 80 channels, 3 layers
6. MBConv6, kernel 5x5, 14x14 resolution, 112 channels, 3 layers
7. MBConv6, kernel 5x5, 14x14 resolution, 192 channels, 4 layers
8. MBConv6, kernel 3x3, 7x7 resolution, 320 channels, 1 layer

9. Conv1x1 & Pooling & Fully Connected, 7x7 resolution, 1280 channels, 1 layer

7.2 Custom Hinge Loss

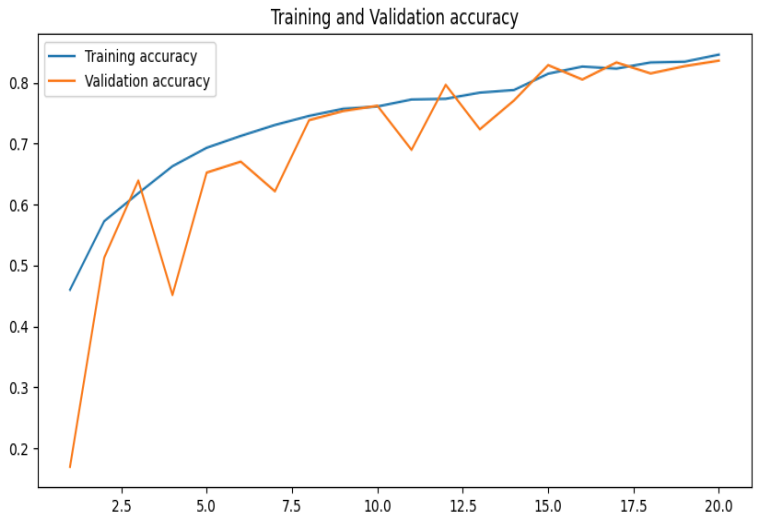
$$L(\mathbf{w}, \mathbf{x}, y) = \max(0, -y(\mathbf{w} \cdot \mathbf{x}))$$

Besides the available loss function in model.compile, Hinge Loss is defined to use as a customize loss function for other choices.

7.3 Training Details

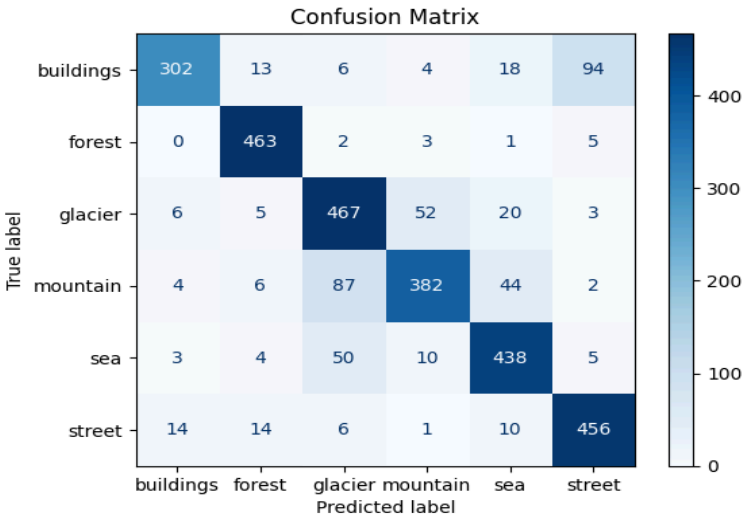
- Loss function: Custom Hinge Loss
- Optimizer: RMSprop
- Metrics: Accuracy
- Epochs: 20
- Batch Size: 32

7.4 Results



Both training and validation accuracy improve steadily, reaching over 80%, indicating good learning and generalization.

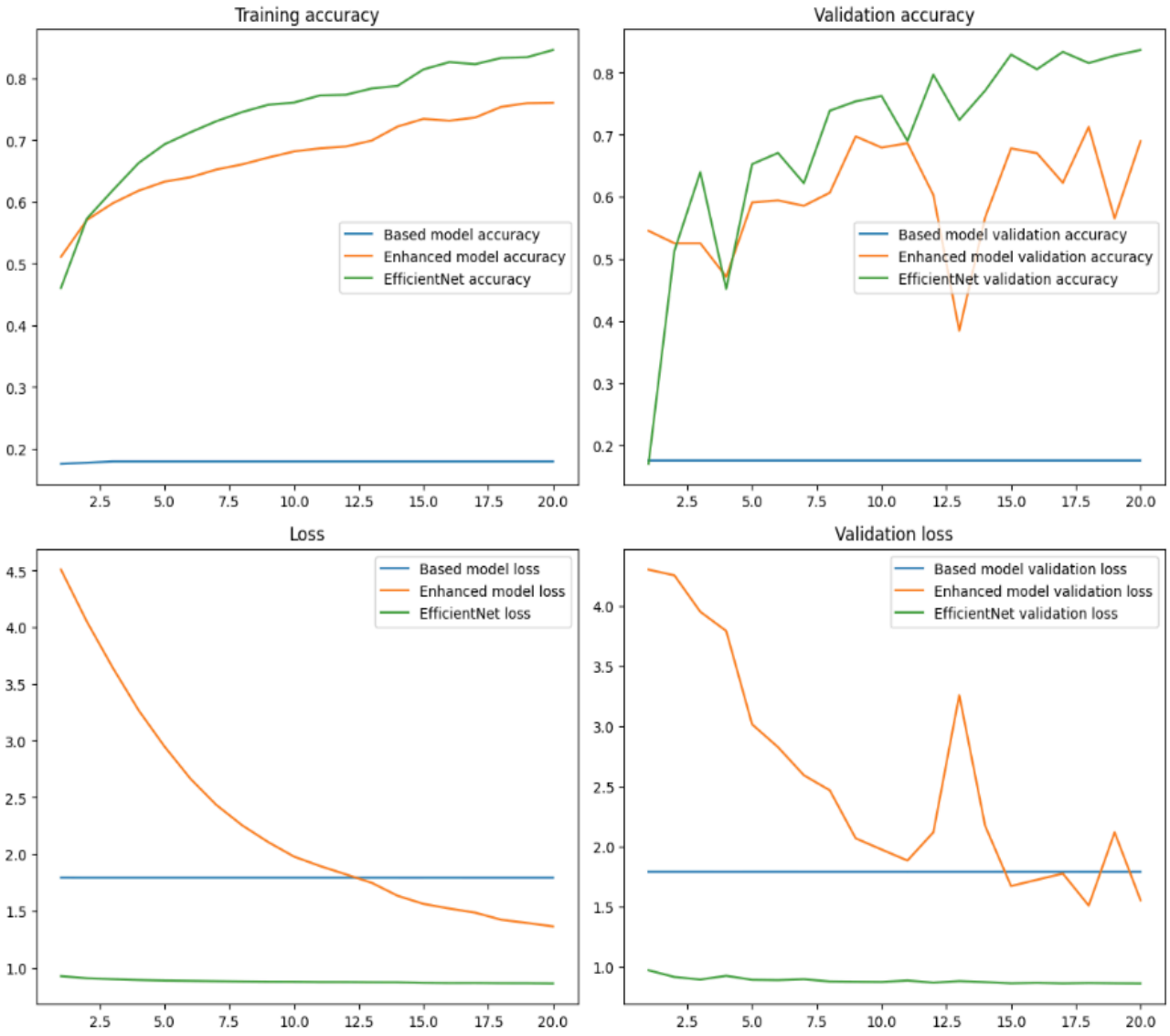
Training and validation loss decrease consistently and stay close, showing no signs of overfitting.



Most predictions are correct. Some confusion occurs between similar classes like buildings vs. street and glacier vs. sea.

8. Comparison

8.1 Accuracy and Loss



Base model: Poor accuracy (~0.17), high and flat loss – clearly underfitting.

Enhanced model: Decent improvement (~0.75 accuracy), but validation metrics are unstable → signs of overfitting.

EfficientNet: Best performer – training and validation accuracy both exceed 0.80, with low, stable loss → good generalization.

8.2 Precision, F1, Recall

Metric	Base Model	Enhanced Model	EfficientNet
Precision	0.0306	0.7591	0.8408
Recall	0.1750	0.7427	0.8360
F1 Score	0.0521	0.7428	0.8341

Base model: Extremely low performance; ineffective.

Enhanced model: Strong improvement, but slightly unstable.

EfficientNet: Best across all metrics — high accuracy, precision, recall, and F1. Most suitable for deployment.

Answer Question 2:

- EfficientNet outperform Base and Enhanced Model achieve higher accuracy as well as lower loss during training process

Advantages of the EfficientNet:

A. Architecture:

- Efficient: Depthwise conv + expansion reduces parameters and computation.
- Powerful features: Swish activations + SE blocks improve feature learning.
- Residuals: Better gradient flow and convergence.
- Scalable: Easily extended (e.g., EfficientNet-B0 → B7).

B. Regularization:

- Global average pooling + dropout = less overfitting
- BatchNorm stabilizes training.

Disadvantages:

A. Complexity:

- Slower training: Due to depth, swish, and SE.
- Harder to tune: More hyperparameters to optimize.
- Less interpretable: Complex block interactions.

B. Deployment:

- Requires more compute power for inference.