Technical report

Deep Learning Approach for detecting nature images

My model is to predict which classification for specific image and I trained a cnn model , VGG16 , VGG19 , RESNET , INCEPTION with data agumentation and I take the high accuracy and I saved the model .h5 and I link this with my ui using pyqt5.

-Training data set :

2191 class buildings

2271 class forests

2404 class glacier

2512 class mountain

2274 class sea

2382 class street

-Testing data set :

437 class buildings

474 class forests

553 class glacier

525 class mountain

510 class sea

501 class street

Deep Learning Models

Our proposed deep learning models use convolutional neural networks (CNNs) to classify **(Sea , Mountain , Buildings , Forest , Street , Glacies)** These CNN models consist of three main types of layers: convolutional, pooling, and fully connected.

- **Convolutional Layers**: Extract features from specific areas of the image and apply an activation function to introduce non-linearity.
- **Pooling Layers**: Reduce the size of the hidden layers and the number of training parameters using techniques like average pooling and max pooling.
- **Fully Connected Layers**: Flatten the features from the pooling layers and pass them through fully connected layers to produce the final classification.

The unique arrangement of these layers allows for efficient feature extraction and robust classification. Additionally, batch normalization and dropout layers can be added to speed up training and minimize Overfitting.

Experimental Results

Evaluation of Deep Learning Models To evaluate the performance of our deep learning models, We used transfer learning to train and use several Convolution-based neural deep learning architectures. In particular, We applied RasNet152v2, Inception, and DensNet. We used ImageNet's pre-trained weights, However, "We added extra layers to the final convolutional layer using the following hyperparameters.": (a) AveragePooling 2D layer 4x4 pixel window, (b) Flattened Layer, (c) Blocks of Rectified Linear

Deep Learning Model Implementation

Our deep learning model utilizes a convolutional neural network (CNN) architecture for classifying images six categories. The model includes several key components:

- **ReLU Activation Function**: Introduces non-linearity.
- **Dropout Layer (70%)**: Reduces overfitting by randomly dropping neurons during training.
- **Linear Layer**: Connects the flattened features to the output layer.
- **Batch Normalization (Batch Norm1D)**: Normalizes the output to improve training speed and stability.
- **Tanh and relu** : to help model in training and fit the output.
- **Sigmoid Function/SoftMax**: Used in the final linear block for binary classification/mnulticlass.

The model is optimized using the Adam optimizer , batch size of **64** images, and trained over 50 epochs. Implemented in TensorFlow **version 2.17.0**, training is conducted on a Jupyter Notebook with GPU support from a T4 GPU.

The dataset is split as follows:

- **for Training** : I choose 600 from 14034 images
- **for Validation** : I choose 600 from 2993 images
- **For Testing** : I choose 7301 from 7301 images

Model performance is evaluated using several metrics: accuracy, precision, recall (sensitivity), and F1 score, ensuring comprehensive assessment of the classification capability.

Network	Accuracy	Recall	Precision	F1-Score
my CNN model	70 %	0.61	0.64	0.59
VGG16	39 %	0.16	0.02	0.04
VGG19	35 %	0.16	0.02	0.04
Inception_V3	32 %	0.16	0.02	0.04
RasNet152V2	45 %	0.16	0.02	0.04
Data augmentation	rotation_range=40, width_shift_range= 0.2, height_shift_range=0.2, shear_range=0.2, zoom_range=0.2,	rotation_range=20, width_shift_range=0.2, height_shift_range=0.2, shear_range=0.2, zoom_range=0.2,	rotation_range=20, width_shift_range=0.2, height_shift_range=0.2, shear_range=0.2, zoom_range=0.2,	rotation_range=20, width_shift_range=0.2, height_shift_range=0.2, shear_range=0.2, zoom_range=0.2,

Conclusion

This is my model and my app that take a photo and predict which class is belong to .
Thank you .