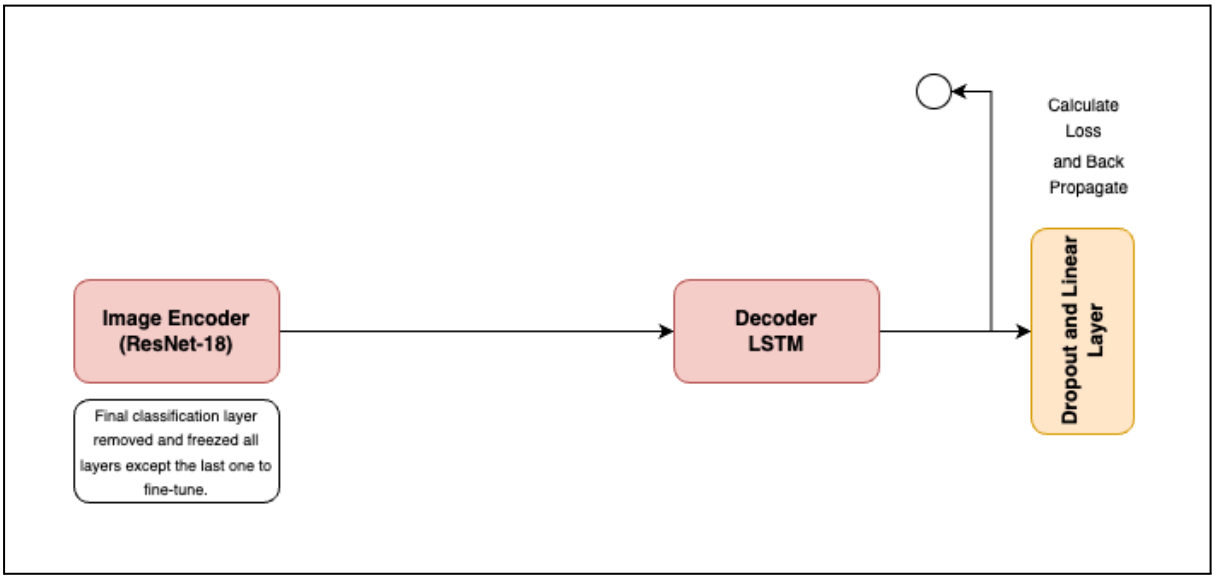
TEAM MEMBERS : ID - 32

23CS91R06	Naquee Rizwan
23CS60R46	Rohan Rajendra Khole
23CS60R41	Atharva Vaidya
23CS60R79	Tushank Panchal

PART - A



Followed Architecture : Used ResNet as an encoder and LSTM as a decoder. Details are present in **MODEL COMPONENTS.**

APPROACH:

This code implements an image captioning model using a combination of Convolutional Neural Network (CNN) for image feature extraction and Recurrent Neural Network (RNN) for generating captions.

CODING COMPONENTS:

Imports: The code begins by importing necessary libraries including pandas for data manipulation, re for regular expressions, string for string operations, NLTK for tokenization, os for file operations, PIL for image handling, and PyTorch for building and training neural networks.

Device Selection: It checks whether a GPU is available and sets the device accordingly.

Data Loading: It loads image filenames and captions from CSV files for training, testing, and validation datasets.
Vocabulary Building: It preprocesses captions to build the vocabulary by tokenizing, converting to lowercase, and filtering out unwanted characters.
Image Preprocessing: It defines image transformations using torchvision.transforms.Compose, which includes resizing, converting to tensor, and normalisation.
Dataset Definition: It defines a custom dataset class (CaptionDataset) inheriting from torch.utils.data.Dataset, responsible for loading images and captions and preprocessing them.

MODEL COMPONENTS:

Encoder (CNN): It uses a pre-trained ResNet-18 model as the CNN encoder for extracting image features. It removes the final classification layer and freezes all layers except the last one to fine-tune.
Decoder (RNN): It defines an RNN-based decoder that takes image features and generates captions. It utilises an embedding layer, an LSTM layer, and a linear layer followed by a dropout layer.
Encoder-Decoder Model: It combines the encoder and decoder into a single model (EncoderDecoder).
Loss Function: It uses cross-entropy loss for computing the loss between predicted and target captions.
Optimizer and Scheduler: It uses the Adam optimizer for optimization and reduces the learning rate on a plateau in the validation loss.
Training Loop: It iterates over epochs, batches, and data loaders for training the model. It computes the loss, performs backpropagation, and updates model parameters. It also evaluates the model on the validation set after each epoch.
Reporting: It prints the training and validation losses for each epoch and adjusts the learning rate based on validation loss.

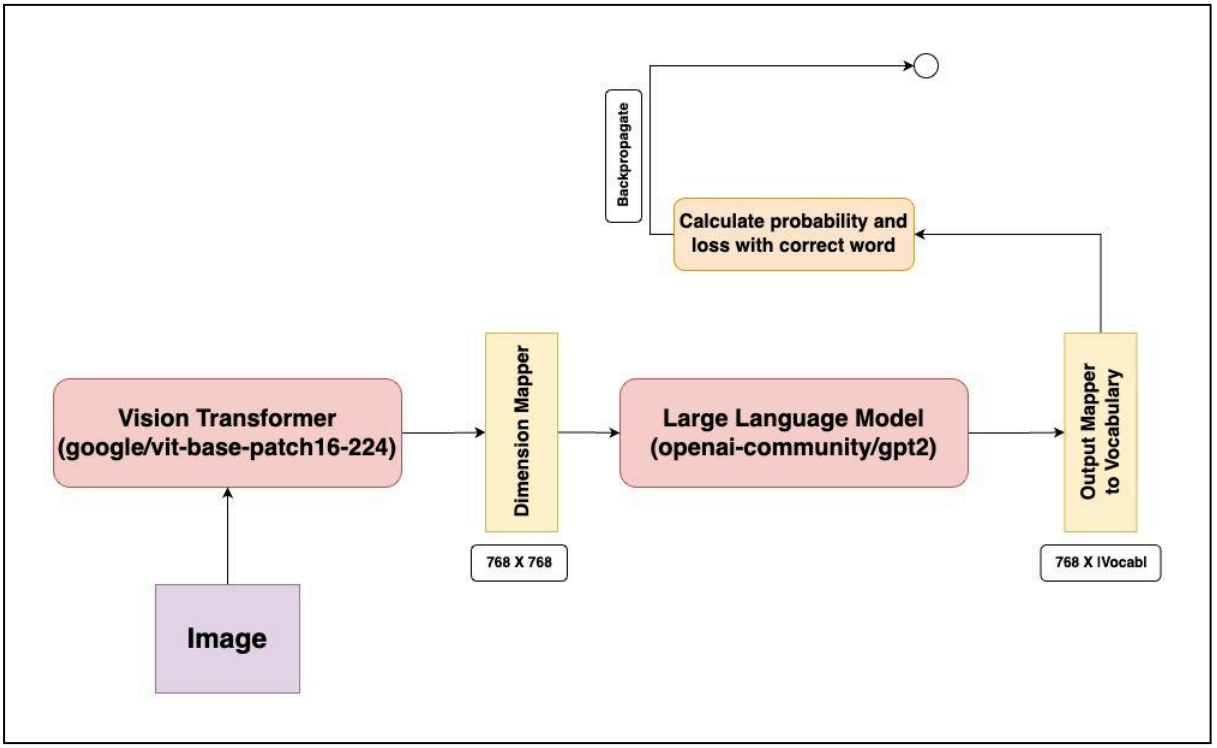
SUMMARY:

In summary, the code constructs an image captioning model by combining CNN for image feature extraction and RNN for caption generation. It's trained using a dataset of images and corresponding captions, with the goal of generating accurate and meaningful captions for unseen images.

EVALUATION METRICS:

BLEU_1	ROUGE_L	CIDEr
0.265	0.263	8.2

PART - B



Followed Architecture : Used Vision Transformer as an encoder and GPT 2 from Hugging face as a decoder. Details are present in **MODEL COMPONENTS.**

APPROACH:

This code implements an image captioning model using a combination of Vision Transformer as encoder and Decoder only LLM as the language decoder. Model checkpoints are shown in the above diagram.

CODING COMPONENTS:

Imports: The code begins by importing necessary libraries mainly including Hugging Face transformers Library. Rest all the libraries in the code are famous Machine Learning Libraries and are self-explanatory.

Device Selection: It checks whether a GPU (“cuda:0”) is available and sets the device accordingly.

Data Loading: It loads image filenames and captions from CSV files for training, testing, and validation datasets.

Image Preprocessing: It defines image transformations using torchvision.transforms.Compose, which includes resizing, converting to tensor, and normalisation.

Dataset Definition: It defines a custom dataset class (CustomCaptionDataset) inheriting from torch.utils.data.Dataset, responsible for loading images and captions and preprocessing them. Along-with this, DataLoader is also used.

MODEL COMPONENTS:

Encoder (Vision Transformer): Used checkpoint is (google/vit-base-patch16-224) from Hugging Face. It takes as input an image, divides it into 16X16 patches and outputs the feature vector worth 768 dimensions. Before feeding this feature vector to LLM, we employ a projection layer to project from Vision embeddings to LLM input format.
Decoder (RNN): We used (`openai-community/gpt2`) as the LLM. It is a decoder only LLM. Its output was in the form of (768X768) which we then changed to (768X|Vocab Size|) so that we get the most suitable word. Softmax and a custom loss function was applied.
Encoder-Decoder Model: It combines the encoder and decoder into a single model (EncoderDecoder) as shown in the above figure.
Loss Function: It uses a custom cross-entropy loss for computing the loss between predicted and target captions. This was needed because, first of all fetched the most probable
Optimizer and Scheduler: It uses the AdamW optimizer for optimization, ExponentialLR as a scheduler.
Training Loop: It iterates over epochs = 1, batches = 2, and data loaders for training the model. It computes the loss, performs backpropagation, and updates model parameters.
Reporting: It prints the training losses for each epoch and adjusts the learning rate based on validation loss.

SUMMARY:

In summary, the code constructs an image captioning model by combining ViT for image feature extraction and LLM for caption generation. It's trained using a dataset of images and corresponding captions, with the goal of generating accurate and meaningful captions for unseen images.

EVALUATION METRICS:

BLEU_1	ROUGE_L	CIDEr
0.405	0.395	24.82

NOTE: We ran all our evaluation metrics using the following Github Page.
https://github.com/Aldenhovel/bleu-rouge-meteor-cider-spice-eval4imagecaption

As it was not feasible to put this gigantic code in a notebook, we mention it up here.