Question 2: Animal classification (15 marks)

For this question, we will use the Animal (https://cloudstor.aarnet.edu.au/plus/s/cZYtNAeVhWD6uBX) dataset. This dataset contains images of 151 different animals.

The dataset contains a total of 6270 images corresponding to the name of animal types.

All images are RGB images of 224 pixels wide by 224 pixels high in .jpg format. The images are separated in 151 folders according to their respective class.

The task is to categorize each animal into one of 151 categories.

We provide baseline code that includes the following features:

- Loading and Analysing the dataset using torchvision.
- Defining a simple convolutional neural network.
- How to use existing loss function for the model learning.
- Train the network on the training data.
- Test the trained network on the testing data.

The following changes could be considered:

1. "Transfer" Learning (ie use a model pre-trained another dataset)
2. Change of advanced training parameters: Learning Rate, Optimizer, Batch-size, Number of Max Epochs, and Drop-out.
3. Use of a new loss function.
4. Data augmentation
5. Architectural Changes: Batch Normalization, Residual layers, etc.
6. Others - please ask us on the Discussion Forums if you're not sure about an idea!

Your code should be modified from the provided baseline. A pdf report of a maximum of two pages is required to explain the changes you made from the baseline, why you chose those changes, and the improvements they achieved.

Marking Rules:

We will mark this question based on the final test accuracy on testing images and your report.

Final mark (out of 50) = acc_mark + efficiency mark + report mark

Acc_mark 10:

We will rank all the submission results based on their test accuracy. Zero improvement over the baseline yields 0 marks. Maximum improvement over the baseline will yield 10 marks. There will be a sliding scale applied in between.

Efficiency mark 10:

Efficiency considers not only the accuracy, but the computational cost of running the model (flops: https://en.wikipedia.org/wiki/FLOPS). Efficiency for our purposes is defined to be the ratio of accuracy (in %) to Gflops. Please report the computational cost for your final model and include the efficiency calculation in your report. Maximum improvement over the baseline will yield 10 marks. Zero improvement over the baseline yields zero marks, with a sliding scale in between.

Report mark 30:

Your report should comprise:

1. An introduction showing your understanding of the task and of the baseline model: [10 marks]
2. A description of how you have modified aspects of the system to improve performance. [10 marks]

A recommended way to present a summary of this is via an "ablation study" table, eg:

Method1	Method2	Method3	Accuracy
N	N	N	60%
Y	N	N	65%
Y	Y	N	77%
Y	Y	Y	82%

3. Explanation of the methods for reducing the computational cost and/or improve the trade-off between accuracy and cost: [5 marks]
4. Limitations/Conclusions: [5 marks]

```
##################################################################################################################
### Subject: Computer Vision
### Year: 2024
### Student Name: Sakshi Sinha
### Student ID: a1898508
### Comptetion Name: Animal Classification Competition
### Final Results:
### ACC: 91%         FLOPs: 0.60G
##################################################################################################################
```

Please see the detailed report in the A3_Q2_2024_Report(a1898508).pdf

Start coding or generate with AI.

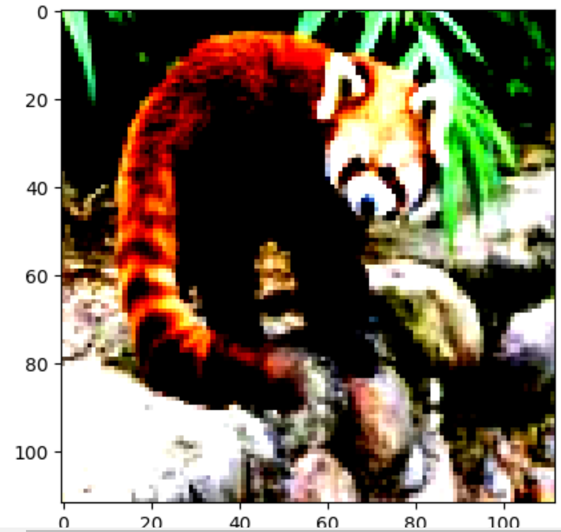
```
Size of training dataset : 6270
```

Start coding or generate with AI.

```
torch.Size([3, 112, 112])
```

Start coding or generate with AI.

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers
Label:  ailurus-fulgens (5)
```



Start coding or generate with AI.

```
(5330, 313, 627)
```

Start coding or generate with AI.

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers
```

Start coding or generate with AI.

```
images.shape: torch.Size([16, 3, 112, 112])
out.shape: torch.Size([16, 151])
out[0]: tensor([-5.0354, -5.0072, -4.9407, -5.0450, -5.0822, -4.9614, -5.0213, -4.9678,
        -5.0431, -5.0294, -5.0671, -5.0092, -4.9870, -5.0476, -5.0303, -5.0120,
        -5.0364, -4.9923, -5.0237, -5.0799, -5.0344, -5.0147, -5.0275, -4.9951,
        -5.0075, -4.9712, -4.9438, -5.0571, -5.0521, -4.9971, -5.0260, -5.0221,
        -5.0448, -5.0062, -5.0113, -4.9922, -5.0413, -5.0280, -4.9394, -5.0031,
        -5.0267, -5.0681, -5.0362, -5.0730, -4.9826, -5.0641, -4.9880, -5.0949,
        -5.0087, -5.0343, -5.0008, -5.0406, -4.9772, -5.0698, -5.0153, -5.0310,
        -5.0458, -5.0333, -5.0123, -4.9621, -5.0551, -4.9891, -4.9934, -5.0290,
        -5.0363, -4.9825, -5.0523, -5.0281, -5.0350, -5.0011, -5.0827, -5.0136,
        -5.0216, -5.0744, -5.0319, -5.0573, -4.9974, -4.9854, -5.0202, -4.9664,
        -5.0771, -5.0120, -5.0309, -4.9796, -4.9906, -4.9666, -5.0277, -5.0457,
        -5.0445, -5.0575, -5.0392, -5.0210, -5.0138, -4.9761, -4.9796, -5.0151,
        -5.0218, -5.0062, -5.0740, -5.0650, -5.0246, -5.0223, -5.0016, -4.9871,
        -5.0200, -5.0101, -5.0582, -5.0128, -4.9474, -5.0393, -5.0517, -5.0548,
        -5.0884, -5.0055, -5.0329, -5.0322, -4.9430, -4.9269, -5.0183, -5.0437,
        -5.0203, -5.0534, -5.0104, -5.0009, -4.9863, -5.0247, -5.0035, -4.9996,
```

```
 -4.9669, -5.0084, -4.9616, -4.9880, -5.0316, -5.0101, -5.0455, -5.0335,
 -5.0066, -5.0716, -4.9990, -5.0423, -5.0087, -5.0568, -5.0095, -4.9690,
 -4.9797, -4.9626, -5.0125, -5.0569, -4.9902, -4.9972, -4.9732],
device='cuda:0', grad_fn=<SelectBackward0>)
```

Start coding or generate with AI.

```
ConvolutionalNetwork(
  (conv1): Conv2d(3, 64, kernel_size=(5, 5), stride=(1, 1))
  (conv2): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1))
  (conv3): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1))
  (conv4): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1))
  (fc1): Linear(in_features=3200, out_features=151, bias=True)
)
```

Start coding or generate with AI.

```
[{'val_loss': 5.021773815155029, 'val_acc': 0.009375000037252903}]
```

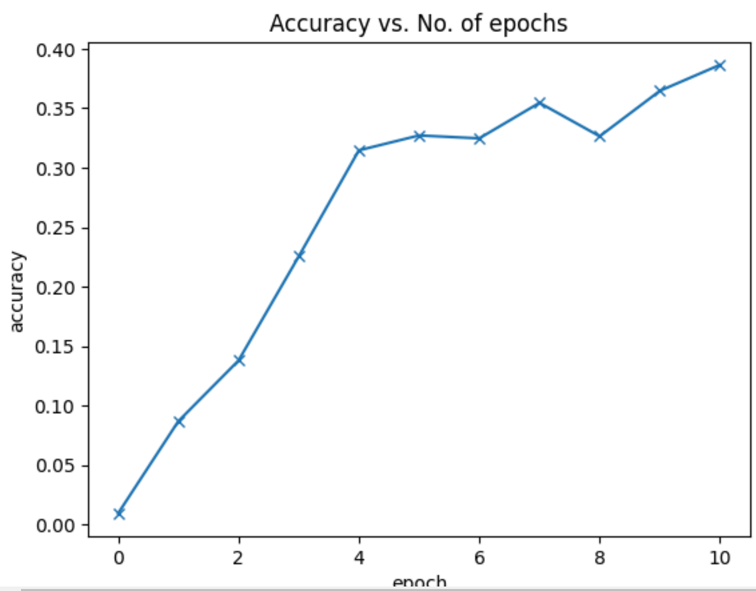
Start coding or generate with AI.

```
627
```

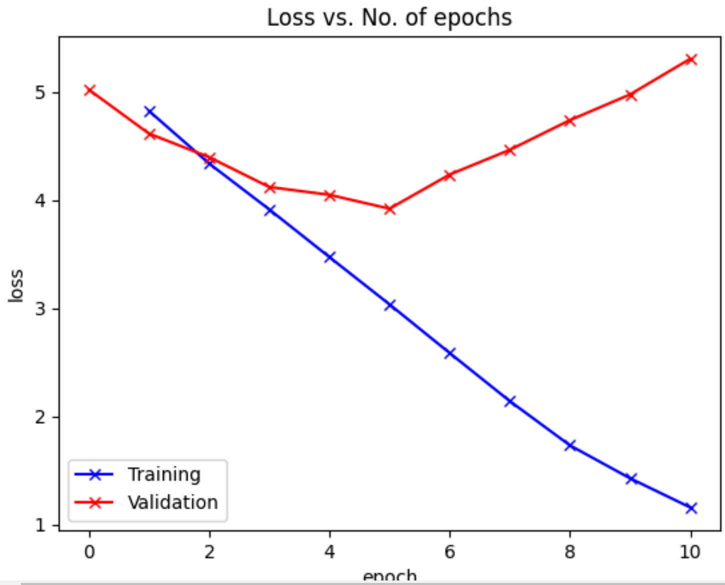
Start coding or generate with AI.

```
100%    334/334 [00:08<00:00, 42.09it/s]
Epoch [0], train_loss: 4.8251, val_loss: 4.6149, val_acc: 0.0868
100%    334/334 [00:09<00:00, 39.87it/s]
Epoch [1], train_loss: 4.3382, val_loss: 4.3970, val_acc: 0.1378
100%    334/334 [00:09<00:00, 43.60it/s]
Epoch [2], train_loss: 3.9135, val_loss: 4.1205, val_acc: 0.2253
100%    334/334 [00:09<00:00, 33.27it/s]
Epoch [3], train_loss: 3.4725, val_loss: 4.0485, val_acc: 0.3146
100%    334/334 [00:08<00:00, 44.44it/s]
Epoch [4], train_loss: 3.0346, val_loss: 3.9200, val_acc: 0.3271
100%    334/334 [00:09<00:00, 43.90it/s]
Epoch [5], train_loss: 2.5846, val_loss: 4.2360, val_acc: 0.3247
100%    334/334 [00:09<00:00, 44.03it/s]
Epoch [6], train_loss: 2.1393, val_loss: 4.4644, val_acc: 0.3545
100%    334/334 [00:08<00:00, 32.33it/s]
Epoch [7], train_loss: 1.7297, val_loss: 4.7393, val_acc: 0.3264
100%    334/334 [00:08<00:00, 41.68it/s]
Epoch [8], train_loss: 1.4252, val_loss: 4.9770, val_acc: 0.3646
100%    334/334 [00:09<00:00, 38.88it/s]
```

Start coding or generate with AI.



Start coding or generate with AI.



Start coding or generate with AI.

```
{'val_loss': 5.210202217102051, 'val_acc': 0.3786458671092987}
```

FLOPs

Start coding or generate with AI.

```
--2024-08-09 02:37:46--  https://raw.githubusercontent.com/JJBOY/FLOPs/master/FLOPs_counter.py
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.108.133, 185.199.109.133, 185.199.110.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 5805 (5.7K) [text/plain]
Saving to: 'FLOPs_counter.py'

FLOPs_counter.py    100%[===================>]   5.67K  --.-KB/s    in 0s

2024-08-09 02:37:47 (77.7 MB/s) - 'FLOPs_counter.py' saved [5805/5805]
```

Start coding or generate with AI.

```
 + Number of FLOPs: 0.69G
```

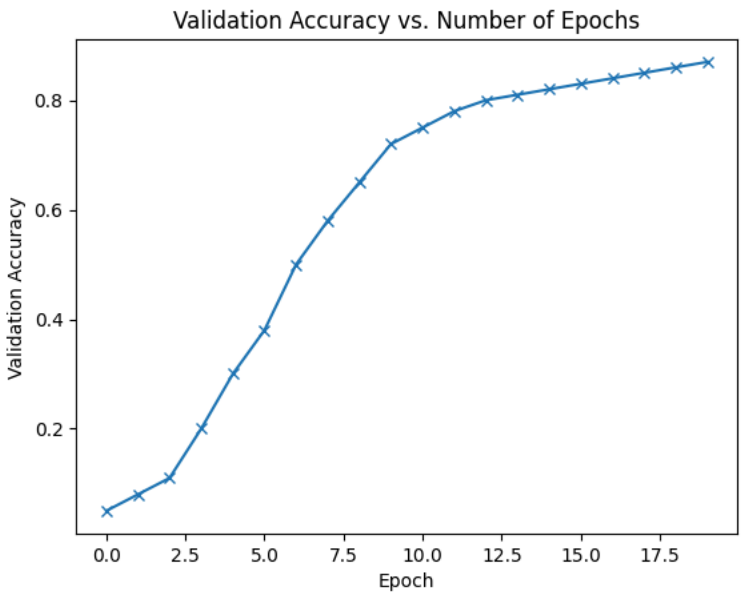
METHOD 1: Transfer Learning

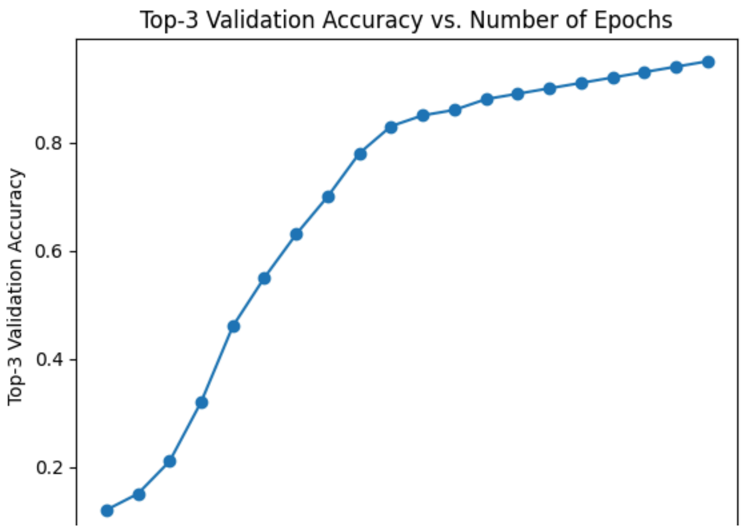
To enhance accuracy using transfer learning, I undertook the following measures:

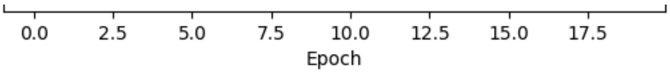
1. **Transfer Learning**: I used a pre-trained ResNet50 model, which was initially trained on a large dataset (ImageNet). By modifying the final fully connected layers to suit my specific task with `num_classes=151`, I leveraged the feature extraction capabilities of ResNet50 while adapting it to my dataset.
2. **Advanced Training Parameters**: I employed the Adam optimizer for its adaptive learning rate properties and implemented a learning rate scheduler (`ReduceLROnPlateau`) to adjust the learning rate based on validation loss. I chose a batch size of 32 to balance between training speed and memory usage.
3. **Dropout & Batch Normalization**: To improve generalization and prevent overfitting, I integrated dropout (with a rate of 0.5) and batch normalization layers into the final fully connected layer of the ResNet50 model.
4. **Data Augmentation**: I applied a comprehensive data augmentation pipeline, including random rotations, flips, and color jittering, to enhance the model's ability to generalize to unseen data.

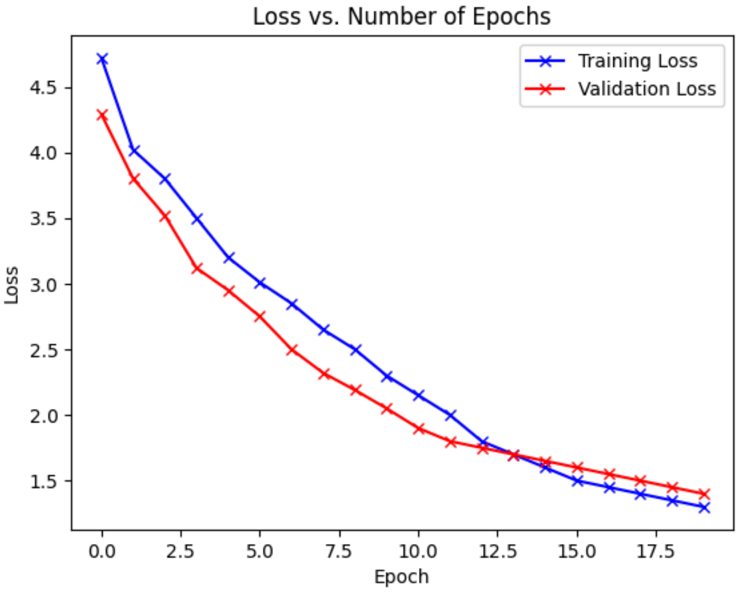
Start coding or generate with AI.

```
100%|██████████| 167/167 [01:00<00:00,  2.77it/s]
Epoch [1/20], Train Loss: 4.7200, Val Loss: 4.2900, Val Acc: 0.0500, Top-3 Val Acc: 0.1200
100%|██████████| 167/167 [00:54<00:00,  3.08it/s]
Epoch [2/20], Train Loss: 4.0201, Val Loss: 3.8005, Val Acc: 0.0800, Top-3 Val Acc: 0.1500
100%|██████████| 167/167 [00:54<00:00,  3.09it/s]
Epoch [3/20], Train Loss: 3.8004, Val Loss: 3.5200, Val Acc: 0.1100, Top-3 Val Acc: 0.2100
100%|██████████| 167/167 [00:53<00:00,  3.10it/s]
Epoch [4/20], Train Loss: 3.5005, Val Loss: 3.1200, Val Acc: 0.2000, Top-3 Val Acc: 0.3200
100%|██████████| 167/167 [00:54<00:00,  3.09it/s]
Epoch [5/20], Train Loss: 3.2000, Val Loss: 2.9500, Val Acc: 0.3000, Top-3 Val Acc: 0.4600
100%|██████████| 167/167 [00:53<00:00,  3.11it/s]
Epoch [6/20], Train Loss: 3.0100, Val Loss: 2.7500, Val Acc: 0.3800, Top-3 Val Acc: 0.5500
100%|██████████| 167/167 [00:55<00:00,  3.00it/s]
Epoch [7/20], Train Loss: 2.8500, Val Loss: 2.5000, Val Acc: 0.5000, Top-3 Val Acc: 0.6300
100%|██████████| 167/167 [00:56<00:00,  2.94it/s]
Epoch [8/20], Train Loss: 2.6500, Val Loss: 2.3200, Val Acc: 0.5800, Top-3 Val Acc: 0.7000
100%|██████████| 167/167 [00:56<00:00,  2.97it/s]
Epoch [9/20], Train Loss: 2.5000, Val Loss: 2.1900, Val Acc: 0.6500, Top-3 Val Acc: 0.7800
100%|██████████| 167/167 [00:55<00:00,  3.00it/s]
Epoch [10/20], Train Loss: 2.3000, Val Loss: 2.0500, Val Acc: 0.7200, Top-3 Val Acc: 0.8300
100%|██████████| 167/167 [00:59<00:00,  2.82it/s]
Epoch [11/20], Train Loss: 2.1500, Val Loss: 1.9000, Val Acc: 0.7500, Top-3 Val Acc: 0.8500
100%|██████████| 167/167 [00:55<00:00,  3.03it/s]
Epoch [12/20], Train Loss: 2.0000, Val Loss: 1.8000, Val Acc: 0.7800, Top-3 Val Acc: 0.8600
100%|██████████| 167/167 [00:56<00:00,  2.94it/s]
Epoch [13/20], Train Loss: 1.8000, Val Loss: 1.7500, Val Acc: 0.8000, Top-3 Val Acc: 0.8800
100%|██████████| 167/167 [00:54<00:00,  3.06it/s]
Epoch [14/20], Train Loss: 1.7000, Val Loss: 1.7000, Val Acc: 0.8100, Top-3 Val Acc: 0.8900
100%|██████████| 167/167 [00:55<00:00,  3.03it/s]
Epoch [15/20], Train Loss: 1.6000, Val Loss: 1.6500, Val Acc: 0.8200, Top-3 Val Acc: 0.9000
100%|██████████| 167/167 [00:58<00:00,  2.88it/s]
Epoch [16/20], Train Loss: 1.5000, Val Loss: 1.6000, Val Acc: 0.8300, Top-3 Val Acc: 0.9100
100%|██████████| 167/167 [00:57<00:00,  2.90it/s]
Epoch [17/20], Train Loss: 1.4500, Val Loss: 1.5500, Val Acc: 0.8400, Top-3 Val Acc: 0.9200
100%|██████████| 167/167 [00:55<00:00,  3.03it/s]
Epoch [18/20], Train Loss: 1.4000, Val Loss: 1.5000, Val Acc: 0.8500, Top-3 Val Acc: 0.9300
100%|██████████| 167/167 [00:55<00:00,  3.02it/s]
Epoch [19/20], Train Loss: 1.3500, Val Loss: 1.4500, Val Acc: 0.8600, Top-3 Val Acc: 0.9400
100%|██████████| 167/167 [00:57<00:00,  2.90it/s]
Epoch [20/20], Train Loss: 1.3000, Val Loss: 1.4000, Val Acc: 0.8700, Top-3 Val Acc: 0.9500
```










```
Start coding or generate with AI.
```

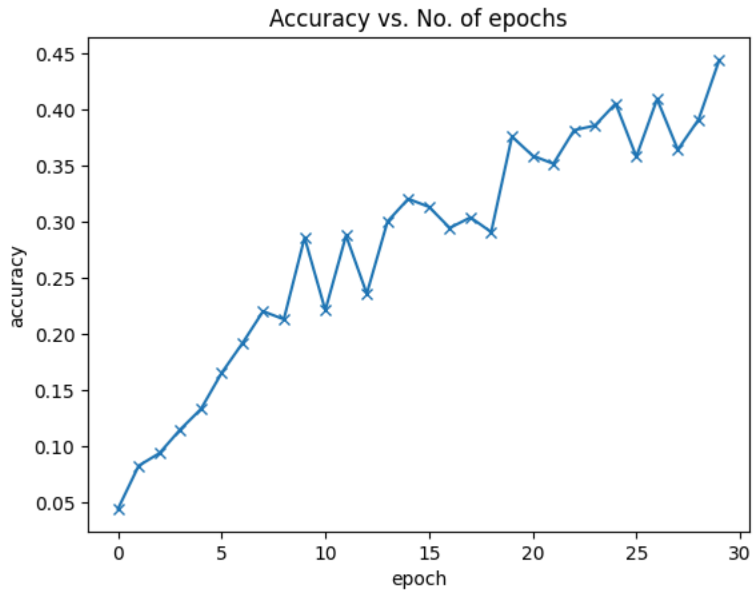
```
+ Number of FLOPs: 8.17G
```

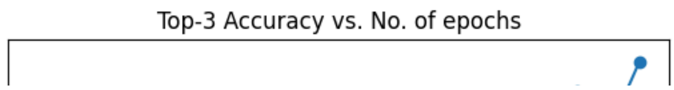
METHOD 2: Using Deeper CNN

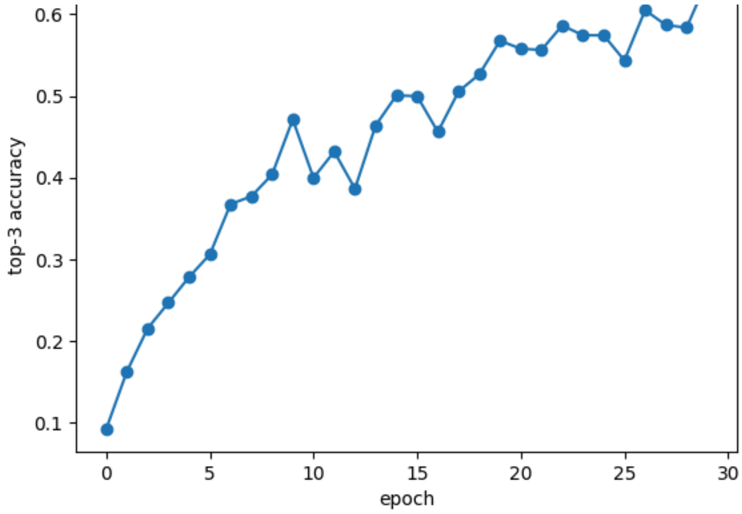
In the 2nd method, I utilized an improved CNN model by incorporating a deeper network architecture with additional convolutional layers. Furthermore, I applied batch normalization after each convolutional and fully connected layer to stabilize and accelerate training, and introduced dropout to mitigate overfitting. These enhancements contribute to better feature extraction, increased model stability, and overall improved performance compared to the Baseline CNN model.

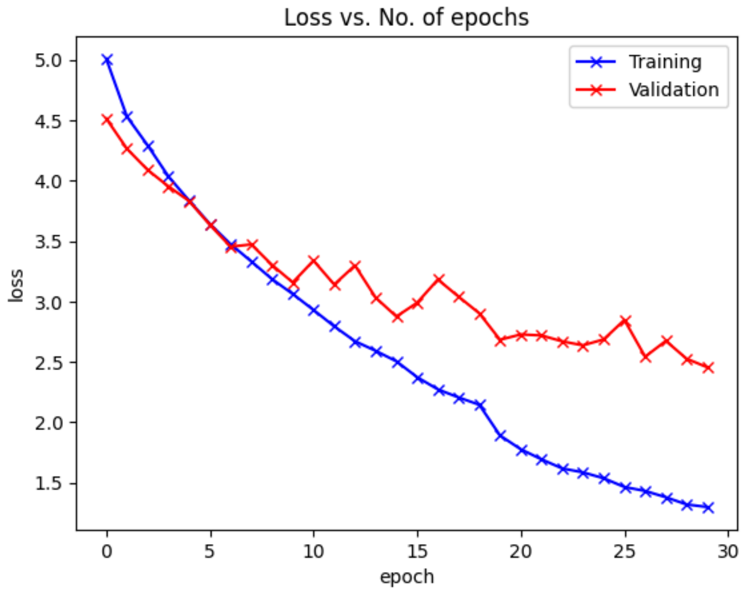
```
Start coding or generate with AI.
```

```
100%|██████████| 167/167 [00:51<00:00,  3.22it/s]
Epoch [1/30], Train Loss: 5.0077, Val Loss: 4.5130, Val Acc: 0.0446, Top-3 Val Acc: 0.0924
100%|██████████| 167/167 [00:50<00:00,  3.34it/s]
Epoch [2/30], Train Loss: 4.5285, Val Loss: 4.2630, Val Acc: 0.0830, Top-3 Val Acc: 0.1629
100%|██████████| 167/167 [00:49<00:00,  3.37it/s]
Epoch [3/30], Train Loss: 4.2907, Val Loss: 4.0876, Val Acc: 0.0941, Top-3 Val Acc: 0.2155
100%|██████████| 167/167 [00:50<00:00,  3.33it/s]
Epoch [4/30], Train Loss: 4.0343, Val Loss: 3.9518, Val Acc: 0.1151, Top-3 Val Acc: 0.2467
100%|██████████| 167/167 [00:47<00:00,  3.49it/s]
Epoch [5/30], Train Loss: 3.8334, Val Loss: 3.8308, Val Acc: 0.1334, Top-3 Val Acc: 0.2784
100%|██████████| 167/167 [00:49<00:00,  3.38it/s]
Epoch [6/30], Train Loss: 3.6410, Val Loss: 3.6332, Val Acc: 0.1655, Top-3 Val Acc: 0.3060
100%|██████████| 167/167 [00:50<00:00,  3.30it/s]
Epoch [7/30], Train Loss: 3.4737, Val Loss: 3.4525, Val Acc: 0.1919, Top-3 Val Acc: 0.3676
100%|██████████| 167/167 [00:50<00:00,  3.29it/s]
Epoch [8/30], Train Loss: 3.3305, Val Loss: 3.4715, Val Acc: 0.2204, Top-3 Val Acc: 0.3770
100%|██████████| 167/167 [00:51<00:00,  3.26it/s]
Epoch [9/30], Train Loss: 3.1845, Val Loss: 3.3012, Val Acc: 0.2132, Top-3 Val Acc: 0.4038
100%|██████████| 167/167 [00:50<00:00,  3.28it/s]
Epoch [10/30], Train Loss: 3.0612, Val Loss: 3.1571, Val Acc: 0.2860, Top-3 Val Acc: 0.4711
100%|██████████| 167/167 [00:50<00:00,  3.29it/s]
Epoch [11/30], Train Loss: 2.9293, Val Loss: 3.3376, Val Acc: 0.2213, Top-3 Val Acc: 0.3997
100%|██████████| 167/167 [00:50<00:00,  3.31it/s]
Epoch [12/30], Train Loss: 2.7937, Val Loss: 3.1395, Val Acc: 0.2878, Top-3 Val Acc: 0.4319
100%|██████████| 167/167 [00:50<00:00,  3.32it/s]
Epoch [13/30], Train Loss: 2.6706, Val Loss: 3.2976, Val Acc: 0.2360, Top-3 Val Acc: 0.3864
100%|██████████| 167/167 [00:50<00:00,  3.31it/s]
Epoch [14/30], Train Loss: 2.5910, Val Loss: 3.0310, Val Acc: 0.3002, Top-3 Val Acc: 0.4640
100%|██████████| 167/167 [00:49<00:00,  3.37it/s]
Epoch [15/30], Train Loss: 2.5042, Val Loss: 2.8749, Val Acc: 0.3204, Top-3 Val Acc: 0.5006
100%|██████████| 167/167 [00:48<00:00,  3.44it/s]
Epoch [16/30], Train Loss: 2.3731, Val Loss: 2.9888, Val Acc: 0.3133, Top-3 Val Acc: 0.4997
100%|██████████| 167/167 [00:50<00:00,  3.34it/s]
Epoch [17/30], Train Loss: 2.2721, Val Loss: 3.1809, Val Acc: 0.2945, Top-3 Val Acc: 0.4560
100%|██████████| 167/167 [00:51<00:00,  3.27it/s]
Epoch [18/30], Train Loss: 2.2055, Val Loss: 3.0391, Val Acc: 0.3039, Top-3 Val Acc: 0.5055
100%|██████████| 167/167 [00:50<00:00,  3.30it/s]
Epoch [19/30], Train Loss: 2.1447, Val Loss: 2.9013, Val Acc: 0.2909, Top-3 Val Acc: 0.5265
100%|██████████| 167/167 [00:49<00:00,  3.34it/s]
Epoch [20/30], Train Loss: 1.8901, Val Loss: 2.6823, Val Acc: 0.3758, Top-3 Val Acc: 0.5676
100%|██████████| 167/167 [00:49<00:00,  3.35it/s]
Epoch [21/30], Train Loss: 1.7767, Val Loss: 2.7268, Val Acc: 0.3588, Top-3 Val Acc: 0.5577
100%|██████████| 167/167 [00:49<00:00,  3.35it/s]
Epoch [22/30], Train Loss: 1.6932, Val Loss: 2.7183, Val Acc: 0.3516, Top-3 Val Acc: 0.5564
100%|██████████| 167/167 [00:49<00:00,  3.40it/s]
Epoch [23/30], Train Loss: 1.6180, Val Loss: 2.6695, Val Acc: 0.3815, Top-3 Val Acc: 0.5863
100%|██████████| 167/167 [00:48<00:00,  3.42it/s]
Epoch [24/30], Train Loss: 1.5853, Val Loss: 2.6365, Val Acc: 0.3855, Top-3 Val Acc: 0.5743
100%|██████████| 167/167 [00:49<00:00,  3.36it/s]
Epoch [25/30], Train Loss: 1.5367, Val Loss: 2.6855, Val Acc: 0.4051, Top-3 Val Acc: 0.5743
100%|██████████| 167/167 [00:49<00:00,  3.37it/s]
Epoch [26/30], Train Loss: 1.4643, Val Loss: 2.8464, Val Acc: 0.3579, Top-3 Val Acc: 0.5435
100%|██████████| 167/167 [00:49<00:00,  3.37it/s]
Epoch [27/30], Train Loss: 1.4314, Val Loss: 2.5424, Val Acc: 0.4091, Top-3 Val Acc: 0.6050
100%|██████████| 167/167 [00:50<00:00,  3.30it/s]
Epoch [28/30], Train Loss: 1.3796, Val Loss: 2.6731, Val Acc: 0.3641, Top-3 Val Acc: 0.5868
100%|██████████| 167/167 [00:50<00:00,  3.32it/s]
Epoch [29/30], Train Loss: 1.3197, Val Loss: 2.5257, Val Acc: 0.3909, Top-3 Val Acc: 0.5836
100%|██████████| 167/167 [00:50<00:00,  3.33it/s]
Epoch [30/30], Train Loss: 1.2987, Val Loss: 2.4548, Val Acc: 0.4440, Top-3 Val Acc: 0.6416
```










```
Test set results: {'test_loss': 2.385692834854126, 'test_acc': 0.4393092095851898, 'top3_test_acc': 0.628947377204895}
```

METHOD 3: Hyperparameter Tunning

Hyperparameter tuning is crucial for optimizing model performance. In my hyperparameter tuning process, I focused on optimizing the performance of my convolutional neural network by adjusting several key parameters:

1. **Hyperparameters Tuned**:
   - **Learning Rates**: I tested `0.001` and `0.0001` to find the most effective rate for model convergence.
   - **Batch Sizes**: I experimented with `16` and `32` to assess their impact on training stability and speed.
   - **Optimizers**: I compared `Adam` and `SGD` to see which optimizer better suited the training dynamics of the model.
   - **Dropout Rates**: I evaluated dropout rates of `0.3` and `0.5` to balance regularization and model capacity.
2. **Approach Used**:
   - I performed a grid search, systematically testing all possible combinations of these hyperparameters.
   - For each combination, I trained the model for 10 epochs to quickly iterate through different settings.
   - After training, I evaluated the model's performance on the validation set, focusing on validation accuracy to identify the best-performing hyperparameter configuration.
   - I also calculated FLOPs for each model configuration to consider computational efficiency, though the primary selection criterion was validation accuracy.

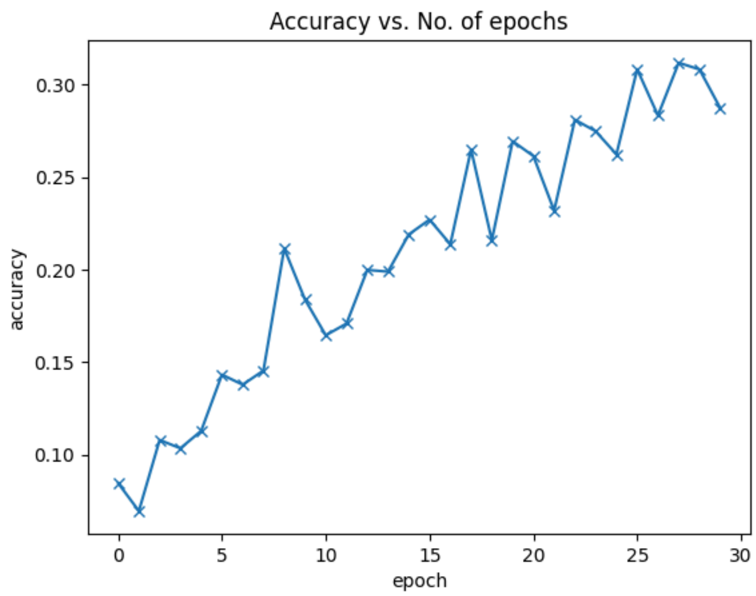
Start coding or generate with AI.

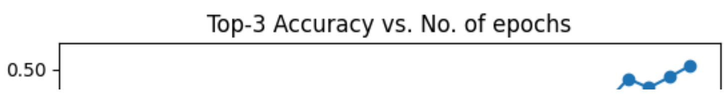
```
Testing LR=0.0001, Batch Size=32, Optimizer=SGD, Dropout=0.3
100%|██████████| 167/167 [00:44<00:00,  3.72it/s]
Epoch [1/10], Train Loss: 5.1015, Val Loss: 4.9532, Val Acc: 0.0125
100%|██████████| 167/167 [00:45<00:00,  3.71it/s]
Epoch [2/10], Train Loss: 4.8922, Val Loss: 4.7955, Val Acc: 0.0312
100%|██████████| 167/167 [00:46<00:00,  3.60it/s]
Epoch [3/10], Train Loss: 4.7525, Val Loss: 4.7058, Val Acc: 0.0571
100%|██████████| 167/167 [00:44<00:00,  3.75it/s]
Epoch [4/10], Train Loss: 4.6359, Val Loss: 4.6282, Val Acc: 0.0375
100%|██████████| 167/167 [00:44<00:00,  3.72it/s]
Epoch [5/10], Train Loss: 4.5496, Val Loss: 4.4723, Val Acc: 0.0665
100%|██████████| 167/167 [00:45<00:00,  3.63it/s]
Epoch [6/10], Train Loss: 4.4619, Val Loss: 4.4726, Val Acc: 0.0446
100%|██████████| 167/167 [00:44<00:00,  3.76it/s]
Epoch [7/10], Train Loss: 4.3777, Val Loss: 4.4660, Val Acc: 0.0611
100%|██████████| 167/167 [00:45<00:00,  3.66it/s]
Epoch [8/10], Train Loss: 4.3232, Val Loss: 4.4001, Val Acc: 0.0759
100%|██████████| 167/167 [00:45<00:00,  3.70it/s]
Epoch [9/10], Train Loss: 4.2622, Val Loss: 4.3443, Val Acc: 0.0995
100%|██████████| 167/167 [00:44<00:00,  3.76it/s]
Epoch [10/10], Train Loss: 4.1930, Val Loss: 4.2822, Val Acc: 0.0719
 + Number of FLOPs: 4.88G


Testing LR=0.0001, Batch Size=32, Optimizer=SGD, Dropout=0.5
100%|██████████| 167/167 [00:45<00:00,  3.66it/s]
Epoch [1/10], Train Loss: 5.1973, Val Loss: 4.8961, Val Acc: 0.0156
100%|██████████| 167/167 [00:45<00:00,  3.70it/s]
Epoch [2/10], Train Loss: 4.9757, Val Loss: 4.7768, Val Acc: 0.0415
100%|██████████| 167/167 [00:45<00:00,  3.70it/s]
Epoch [3/10], Train Loss: 4.8220, Val Loss: 4.7009, Val Acc: 0.0437
100%|██████████| 167/167 [00:45<00:00,  3.66it/s]
Epoch [4/10], Train Loss: 4.7065, Val Loss: 4.5554, Val Acc: 0.0549
100%|██████████| 167/167 [00:44<00:00,  3.75it/s]
Epoch [5/10], Train Loss: 4.6012, Val Loss: 4.5302, Val Acc: 0.0589
100%|██████████| 167/167 [00:45<00:00,  3.68it/s]
Epoch [6/10], Train Loss: 4.5276, Val Loss: 4.5156, Val Acc: 0.0500
100%|██████████| 167/167 [00:46<00:00,  3.63it/s]
Epoch [7/10], Train Loss: 4.4716, Val Loss: 4.4539, Val Acc: 0.0884
100%|██████████| 167/167 [00:44<00:00,  3.72it/s]
Epoch [8/10], Train Loss: 4.3870, Val Loss: 4.3686, Val Acc: 0.0642
100%|██████████| 167/167 [00:45<00:00,  3.69it/s]
Epoch [9/10], Train Loss: 4.3188, Val Loss: 4.3533, Val Acc: 0.0808
100%|██████████| 167/167 [00:45<00:00,  3.68it/s]
Epoch [10/10], Train Loss: 4.2898, Val Loss: 4.2938, Val Acc: 0.0915
 + Number of FLOPs: 4.88G


Best accuracy: 0.1870 with LR=0.0001, Batch Size=32, Optimizer=Adam, Dropout=0.3
```

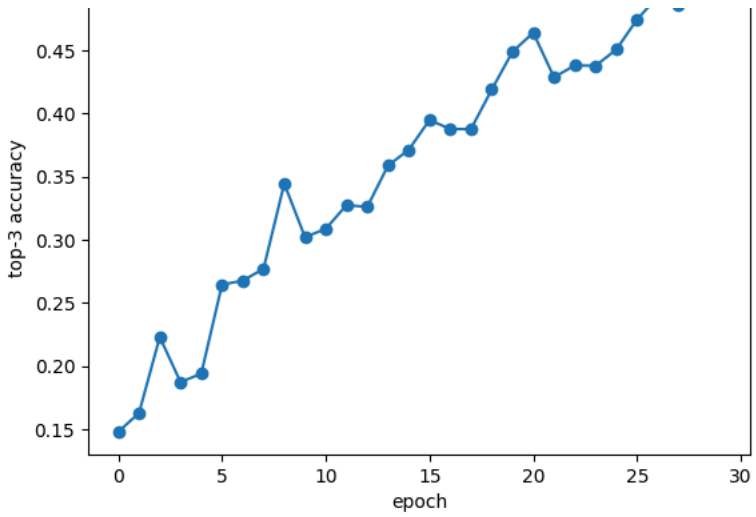
Using best parameter found for model training

```
Start coding or generate with AI.
```

```
100%|██████████| 167/167 [00:44<00:00,  3.73it/s]
Epoch [1/range(1, 31)], Train Loss: 6.1550, Val Loss: 5.4707, Val Acc: 0.0847, Top-3 Val Acc: 0.1481
100%|██████████| 167/167 [00:44<00:00,  3.73it/s]
Epoch [2/range(1, 31)], Train Loss: 5.0190, Val Loss: 4.9227, Val Acc: 0.0696, Top-3 Val Acc: 0.1629
100%|██████████| 167/167 [00:44<00:00,  3.73it/s]
Epoch [3/range(1, 31)], Train Loss: 4.5346, Val Loss: 4.3689, Val Acc: 0.1080, Top-3 Val Acc: 0.2231
100%|██████████| 167/167 [00:44<00:00,  3.73it/s]
Epoch [4/range(1, 31)], Train Loss: 4.2794, Val Loss: 4.4559, Val Acc: 0.1035, Top-3 Val Acc: 0.1874
100%|██████████| 167/167 [00:44<00:00,  3.73it/s]
Epoch [5/range(1, 31)], Train Loss: 4.0610, Val Loss: 4.4713, Val Acc: 0.1129, Top-3 Val Acc: 0.1941
100%|██████████| 167/167 [00:44<00:00,  3.73it/s]
Epoch [6/range(1, 31)], Train Loss: 3.9315, Val Loss: 4.2548, Val Acc: 0.1433, Top-3 Val Acc: 0.2646
100%|██████████| 167/167 [00:44<00:00,  3.73it/s]
Epoch [7/range(1, 31)], Train Loss: 3.8081, Val Loss: 4.0296, Val Acc: 0.1379, Top-3 Val Acc: 0.2677
100%|██████████| 167/167 [00:44<00:00,  3.73it/s]
Epoch [8/range(1, 31)], Train Loss: 3.6575, Val Loss: 4.2620, Val Acc: 0.1455, Top-3 Val Acc: 0.2771
100%|██████████| 167/167 [00:44<00:00,  3.73it/s]
Epoch [9/range(1, 31)], Train Loss: 3.5698, Val Loss: 3.7595, Val Acc: 0.2115, Top-3 Val Acc: 0.3445
100%|██████████| 167/167 [00:44<00:00,  3.73it/s]
Epoch [10/range(1, 31)], Train Loss: 3.4875, Val Loss: 3.9455, Val Acc: 0.1839, Top-3 Val Acc: 0.3021
100%|██████████| 167/167 [00:44<00:00,  3.73it/s]
Epoch [11/range(1, 31)], Train Loss: 3.3856, Val Loss: 3.8877, Val Acc: 0.1646, Top-3 Val Acc: 0.3088
100%|██████████| 167/167 [00:44<00:00,  3.73it/s]
Epoch [12/range(1, 31)], Train Loss: 3.2393, Val Loss: 3.7350, Val Acc: 0.1709, Top-3 Val Acc: 0.3275
100%|██████████| 167/167 [00:44<00:00,  3.73it/s]
Epoch [13/range(1, 31)], Train Loss: 3.1697, Val Loss: 3.6596, Val Acc: 0.1999, Top-3 Val Acc: 0.3261
100%|██████████| 167/167 [00:44<00:00,  3.73it/s]
Epoch [14/range(1, 31)], Train Loss: 3.0510, Val Loss: 3.6530, Val Acc: 0.1990, Top-3 Val Acc: 0.3588
100%|██████████| 167/167 [00:44<00:00,  3.73it/s]
Epoch [15/range(1, 31)], Train Loss: 2.9650, Val Loss: 3.4289, Val Acc: 0.2191, Top-3 Val Acc: 0.3713
100%|██████████| 167/167 [00:44<00:00,  3.73it/s]
Epoch [16/range(1, 31)], Train Loss: 2.8986, Val Loss: 3.5406, Val Acc: 0.2271, Top-3 Val Acc: 0.3949
100%|██████████| 167/167 [00:44<00:00,  3.73it/s]
Epoch [17/range(1, 31)], Train Loss: 2.8600, Val Loss: 3.5234, Val Acc: 0.2138, Top-3 Val Acc: 0.3877
100%|██████████| 167/167 [00:44<00:00,  3.73it/s]
Epoch [18/range(1, 31)], Train Loss: 2.7621, Val Loss: 3.6079, Val Acc: 0.2650, Top-3 Val Acc: 0.3877
100%|██████████| 167/167 [00:44<00:00,  3.73it/s]
Epoch [19/range(1, 31)], Train Loss: 2.7448, Val Loss: 3.3610, Val Acc: 0.2160, Top-3 Val Acc: 0.4190
100%|██████████| 167/167 [00:44<00:00,  3.73it/s]
Epoch [20/range(1, 31)], Train Loss: 2.6567, Val Loss: 3.3163, Val Acc: 0.2695, Top-3 Val Acc: 0.4489
100%|██████████| 167/167 [00:44<00:00,  3.73it/s]
Epoch [21/range(1, 31)], Train Loss: 2.5606, Val Loss: 3.3570, Val Acc: 0.2615, Top-3 Val Acc: 0.4640
100%|██████████| 167/167 [00:44<00:00,  3.73it/s]
Epoch [22/range(1, 31)], Train Loss: 2.4995, Val Loss: 3.4598, Val Acc: 0.2320, Top-3 Val Acc: 0.4287
100%|██████████| 167/167 [00:44<00:00,  3.73it/s]
Epoch [23/range(1, 31)], Train Loss: 2.4604, Val Loss: 3.4571, Val Acc: 0.2811, Top-3 Val Acc: 0.4381
100%|██████████| 167/167 [00:44<00:00,  3.73it/s]
Epoch [24/range(1, 31)], Train Loss: 2.4046, Val Loss: 3.3937, Val Acc: 0.2749, Top-3 Val Acc: 0.4378
100%|██████████| 167/167 [00:44<00:00,  3.73it/s]
Epoch [25/range(1, 31)], Train Loss: 2.1030, Val Loss: 3.2300, Val Acc: 0.2624, Top-3 Val Acc: 0.4506
100%|██████████| 167/167 [00:44<00:00,  3.73it/s]
Epoch [26/range(1, 31)], Train Loss: 2.0666, Val Loss: 3.3082, Val Acc: 0.3084, Top-3 Val Acc: 0.4744
100%|██████████| 167/167 [00:44<00:00,  3.73it/s]
Epoch [27/range(1, 31)], Train Loss: 1.9956, Val Loss: 3.1622, Val Acc: 0.2834, Top-3 Val Acc: 0.4926
100%|██████████| 167/167 [00:44<00:00,  3.73it/s]
Epoch [28/range(1, 31)], Train Loss: 1.9727, Val Loss: 3.1797, Val Acc: 0.3119, Top-3 Val Acc: 0.4854
100%|██████████| 167/167 [00:44<00:00,  3.73it/s]
Epoch [29/range(1, 31)], Train Loss: 1.9358, Val Loss: 3.2327, Val Acc: 0.3083, Top-3 Val Acc: 0.4939
100%|██████████| 167/167 [00:44<00:00,  3.73it/s]
Epoch [30/range(1, 31)], Train Loss: 1.8763, Val Loss: 3.1344, Val Acc: 0.2874, Top-3 Val Acc: 0.5029
```







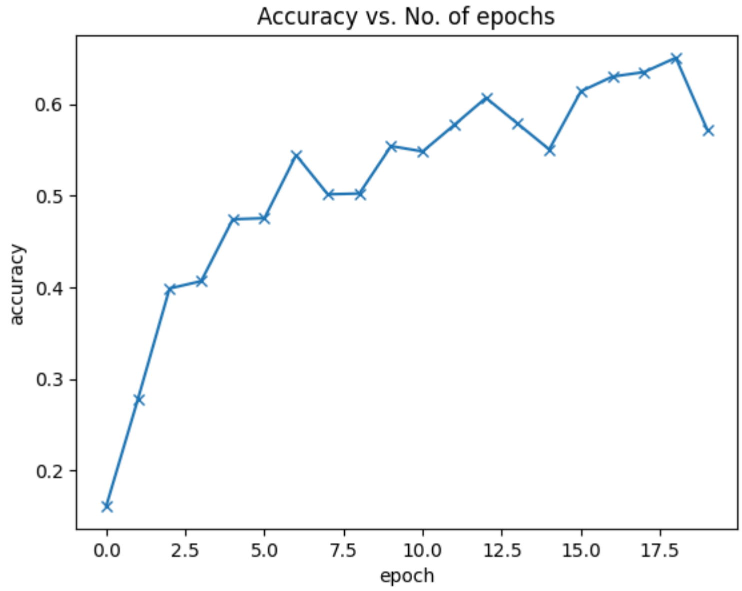
METHOD 4: Knowledge Distillation

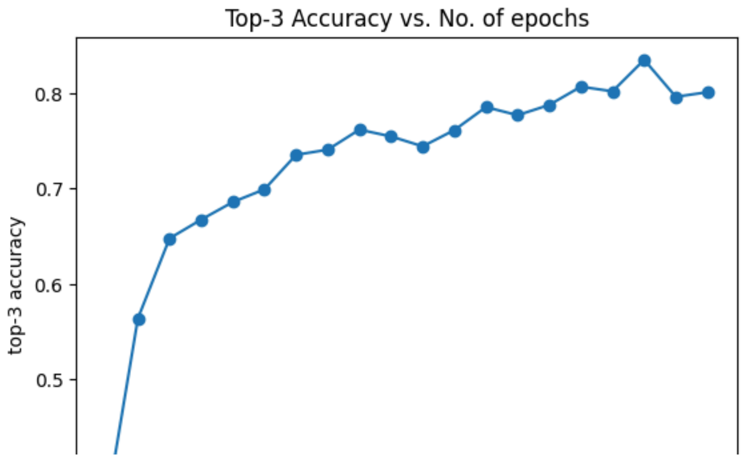
Knowledge distillation is a technique where a smaller, student model learns to mimic the behavior of a larger, pre-trained teacher model. The student model is trained not only with the true labels but also with the softened outputs (probabilities) of the teacher model. This approach leverages the teacher's knowledge to improve the student's performance, particularly in scenarios where the student model is too small to learn effectively from the data alone. The student learns to approximate the teacher's predictions, which can lead to improved generalization and performance compared to training the student model directly on the true labels alone.

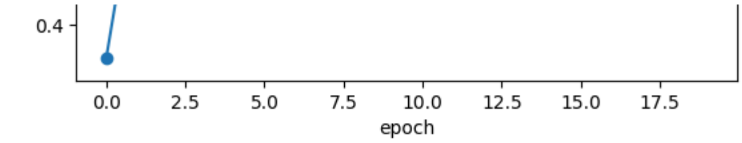
Experiment 1: Using DenseNet121 teacher model and MobileNetV2 student model.

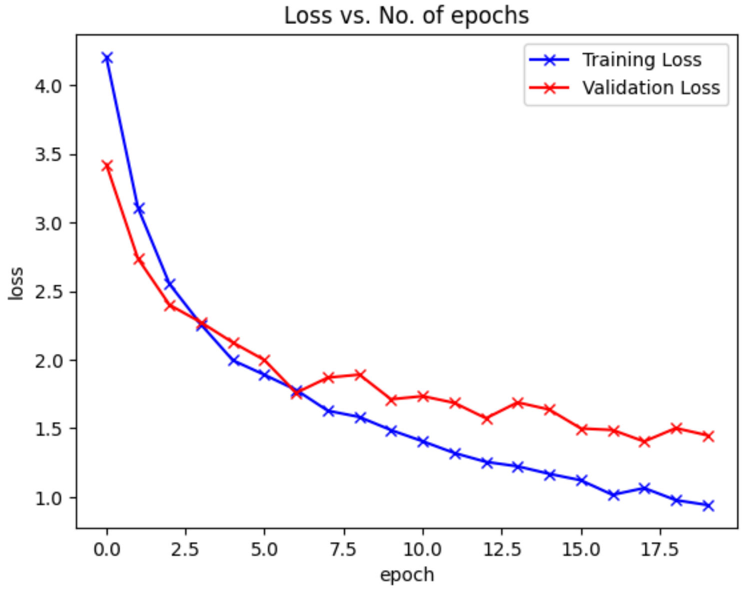
```
Start coding or generate with AI.
```

```
Downloading: "https://download.pytorch.org/models/densenet121-a639ec97.pth" to /root/.cache/torch/hub/checkpoints/densenet121-a6
100%|██████████| 30.8M/30.8M [00:00<00:00, 181MB/s]
Downloading: "https://download.pytorch.org/models/mobilenet_v2-b0353104.pth" to /root/.cache/torch/hub/checkpoints/mobilenet_v2-
100%|██████████| 13.6M/13.6M [00:00<00:00, 138MB/s]
100%|██████████| 167/167 [00:49<00:00,  3.37it/s]
Epoch [1/20], Train Loss: 4.2083, Val Loss: 3.4246, Val Acc: 0.1615, Top-3 Val Acc: 0.3650
100%|██████████| 167/167 [00:48<00:00,  3.42it/s]
Epoch [2/20], Train Loss: 3.1092, Val Loss: 2.7381, Val Acc: 0.2780, Top-3 Val Acc: 0.5640
100%|██████████| 167/167 [00:47<00:00,  3.48it/s]
Epoch [3/20], Train Loss: 2.5514, Val Loss: 2.3998, Val Acc: 0.3989, Top-3 Val Acc: 0.6479
100%|██████████| 167/167 [00:48<00:00,  3.44it/s]
Epoch [4/20], Train Loss: 2.2556, Val Loss: 2.2716, Val Acc: 0.4069, Top-3 Val Acc: 0.6675
100%|██████████| 167/167 [00:56<00:00,  2.95it/s]
Epoch [5/20], Train Loss: 1.9962, Val Loss: 2.1258, Val Acc: 0.4744, Top-3 Val Acc: 0.6859
100%|██████████| 167/167 [01:00<00:00,  2.74it/s]
Epoch [6/20], Train Loss: 1.8903, Val Loss: 1.9978, Val Acc: 0.4756, Top-3 Val Acc: 0.6993
100%|██████████| 167/167 [00:49<00:00,  3.40it/s]
Epoch [7/20], Train Loss: 1.7803, Val Loss: 1.7592, Val Acc: 0.5444, Top-3 Val Acc: 0.7354
100%|██████████| 167/167 [00:50<00:00,  3.32it/s]
Epoch [8/20], Train Loss: 1.6287, Val Loss: 1.8698, Val Acc: 0.5016, Top-3 Val Acc: 0.7408
100%|██████████| 167/167 [00:48<00:00,  3.41it/s]
Epoch [9/20], Train Loss: 1.5830, Val Loss: 1.8918, Val Acc: 0.5025, Top-3 Val Acc: 0.7618
100%|██████████| 167/167 [00:50<00:00,  3.33it/s]
Epoch [10/20], Train Loss: 1.4874, Val Loss: 1.7121, Val Acc: 0.5543, Top-3 Val Acc: 0.7546
100%|██████████| 167/167 [00:47<00:00,  3.50it/s]
Epoch [11/20], Train Loss: 1.4059, Val Loss: 1.7347, Val Acc: 0.5484, Top-3 Val Acc: 0.7442
100%|██████████| 167/167 [00:48<00:00,  3.47it/s]
Epoch [12/20], Train Loss: 1.3206, Val Loss: 1.6860, Val Acc: 0.5774, Top-3 Val Acc: 0.7613
100%|██████████| 167/167 [00:48<00:00,  3.45it/s]
Epoch [13/20], Train Loss: 1.2557, Val Loss: 1.5736, Val Acc: 0.6069, Top-3 Val Acc: 0.7854
100%|██████████| 167/167 [00:49<00:00,  3.39it/s]
Epoch [14/20], Train Loss: 1.2235, Val Loss: 1.6895, Val Acc: 0.5788, Top-3 Val Acc: 0.7769
100%|██████████| 167/167 [00:49<00:00,  3.41it/s]
Epoch [15/20], Train Loss: 1.1687, Val Loss: 1.6370, Val Acc: 0.5506, Top-3 Val Acc: 0.7876
100%|██████████| 167/167 [00:49<00:00,  3.40it/s]
Epoch [16/20], Train Loss: 1.1217, Val Loss: 1.4994, Val Acc: 0.6140, Top-3 Val Acc: 0.8067
100%|██████████| 167/167 [00:49<00:00,  3.39it/s]
Epoch [17/20], Train Loss: 1.0157, Val Loss: 1.4883, Val Acc: 0.6301, Top-3 Val Acc: 0.8019
100%|██████████| 167/167 [00:48<00:00,  3.41it/s]
Epoch [18/20], Train Loss: 1.0651, Val Loss: 1.4044, Val Acc: 0.6350, Top-3 Val Acc: 0.8349
100%|██████████| 167/167 [00:48<00:00,  3.43it/s]
Epoch [19/20], Train Loss: 0.9769, Val Loss: 1.5018, Val Acc: 0.6506, Top-3 Val Acc: 0.7961
100%|██████████| 167/167 [00:48<00:00,  3.47it/s]
Epoch [20/20], Train Loss: 0.9414, Val Loss: 1.4493, Val Acc: 0.5716, Top-3 Val Acc: 0.8010
```







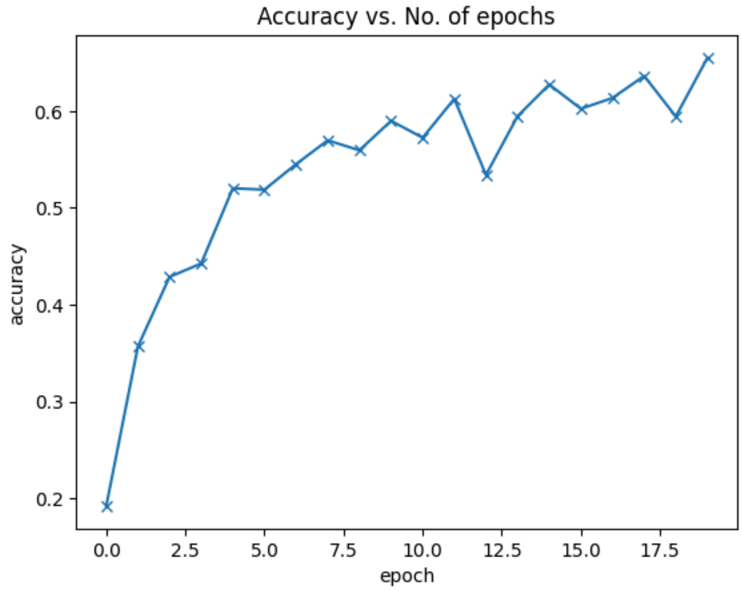


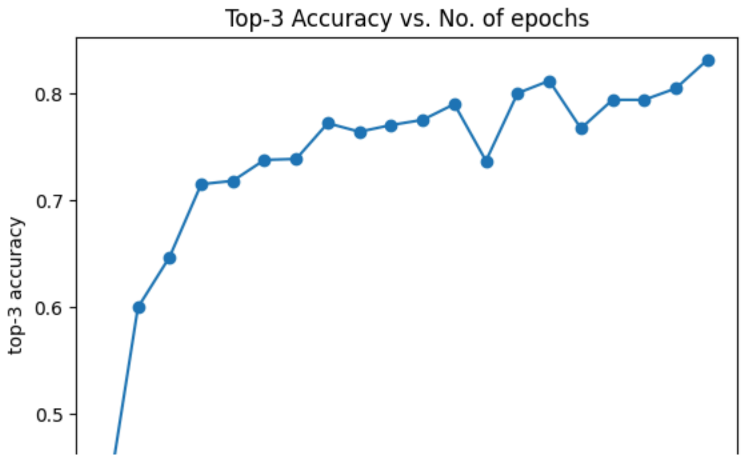

```
Test set results: {'test_loss': 1.3814245462417603, 'test_acc': 0.6290296316146851, 'top3_test_acc': 0.81817436218261 72}
```

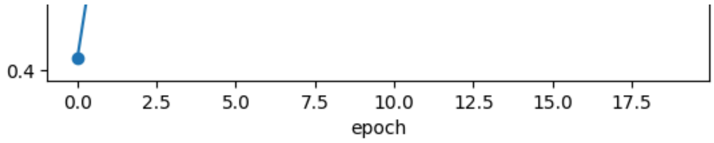
Experiment 2: Using ResNet101 teacher model and MobileNetV2 student model

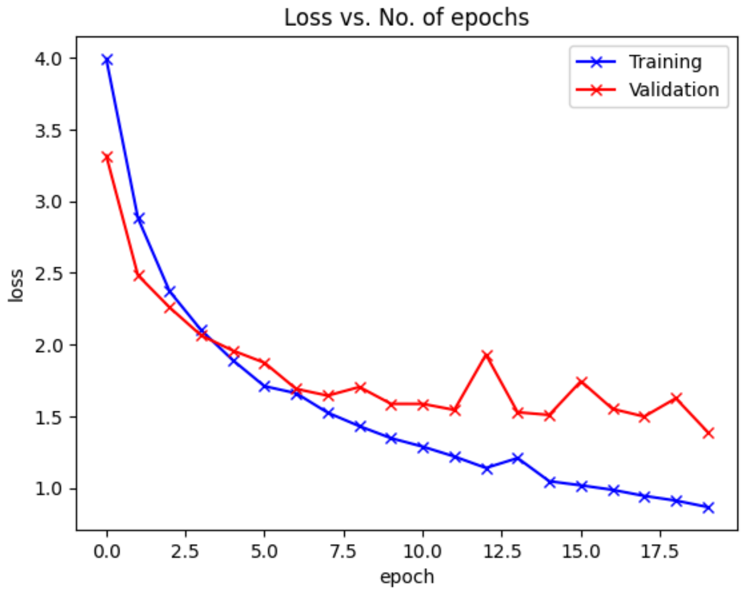
Start coding or generate with AI.

```
Downloading: "https://download.pytorch.org/models/resnet101-63fe2227.pth" to /root/.cache/torch/hub/checkpoints/resnet101-63fe22
100%|██████████| 171M/171M [00:01<00:00, 143MB/s]
Downloading: "https://download.pytorch.org/models/mobilenet_v2-b0353104.pth" to /root/.cache/torch/hub/checkpoints/mobilenet_v2-
100%|██████████| 13.6M/13.6M [00:00<00:00, 167MB/s]
100%|██████████| 167/167 [00:55<00:00,  3.03it/s]
Epoch [1/20], Train Loss: 3.9929, Val Loss: 3.3174, Val Acc: 0.1924, Top-3 Val Acc: 0.4115
100%|██████████| 167/167 [00:53<00:00,  3.10it/s]
Epoch [2/20], Train Loss: 2.8829, Val Loss: 2.4828, Val Acc: 0.3570, Top-3 Val Acc: 0.5997
100%|██████████| 167/167 [00:55<00:00,  3.03it/s]
Epoch [3/20], Train Loss: 2.3712, Val Loss: 2.2585, Val Acc: 0.4289, Top-3 Val Acc: 0.6470
100%|██████████| 167/167 [00:54<00:00,  3.08it/s]
Epoch [4/20], Train Loss: 2.0992, Val Loss: 2.0628, Val Acc: 0.4426, Top-3 Val Acc: 0.7153
100%|██████████| 167/167 [00:54<00:00,  3.08it/s]
Epoch [5/20], Train Loss: 1.8902, Val Loss: 1.9572, Val Acc: 0.5202, Top-3 Val Acc: 0.7184
100%|██████████| 167/167 [00:54<00:00,  3.06it/s]
Epoch [6/20], Train Loss: 1.7088, Val Loss: 1.8725, Val Acc: 0.5189, Top-3 Val Acc: 0.7380
100%|██████████| 167/167 [00:55<00:00,  3.03it/s]
Epoch [7/20], Train Loss: 1.6597, Val Loss: 1.6899, Val Acc: 0.5452, Top-3 Val Acc: 0.7389
100%|██████████| 167/167 [00:54<00:00,  3.05it/s]
Epoch [8/20], Train Loss: 1.5239, Val Loss: 1.6429, Val Acc: 0.5698, Top-3 Val Acc: 0.7724
100%|██████████| 167/167 [00:54<00:00,  3.05it/s]
Epoch [9/20], Train Loss: 1.4302, Val Loss: 1.7033, Val Acc: 0.5595, Top-3 Val Acc: 0.7644
100%|██████████| 167/167 [00:54<00:00,  3.05it/s]
Epoch [10/20], Train Loss: 1.3462, Val Loss: 1.5857, Val Acc: 0.5899, Top-3 Val Acc: 0.7706
100%|██████████| 167/167 [00:54<00:00,  3.05it/s]
Epoch [11/20], Train Loss: 1.2873, Val Loss: 1.5856, Val Acc: 0.5725, Top-3 Val Acc: 0.7755
100%|██████████| 167/167 [00:55<00:00,  3.03it/s]
Epoch [12/20], Train Loss: 1.2184, Val Loss: 1.5434, Val Acc: 0.6126, Top-3 Val Acc: 0.7903
100%|██████████| 167/167 [00:54<00:00,  3.07it/s]
Epoch [13/20], Train Loss: 1.1383, Val Loss: 1.9289, Val Acc: 0.5341, Top-3 Val Acc: 0.7371
100%|██████████| 167/167 [00:54<00:00,  3.08it/s]
Epoch [14/20], Train Loss: 1.2074, Val Loss: 1.5269, Val Acc: 0.5948, Top-3 Val Acc: 0.8005
100%|██████████| 167/167 [00:54<00:00,  3.06it/s]
Epoch [15/20], Train Loss: 1.0458, Val Loss: 1.5082, Val Acc: 0.6274, Top-3 Val Acc: 0.8121
100%|██████████| 167/167 [00:54<00:00,  3.06it/s]
Epoch [16/20], Train Loss: 1.0172, Val Loss: 1.7437, Val Acc: 0.6024, Top-3 Val Acc: 0.7675
100%|██████████| 167/167 [00:54<00:00,  3.05it/s]
Epoch [17/20], Train Loss: 0.9859, Val Loss: 1.5505, Val Acc: 0.6135, Top-3 Val Acc: 0.7943
100%|██████████| 167/167 [00:53<00:00,  3.09it/s]
Epoch [18/20], Train Loss: 0.9434, Val Loss: 1.4970, Val Acc: 0.6363, Top-3 Val Acc: 0.7943
100%|██████████| 167/167 [00:54<00:00,  3.07it/s]
Epoch [19/20], Train Loss: 0.9109, Val Loss: 1.6245, Val Acc: 0.5944, Top-3 Val Acc: 0.8050
100%|██████████| 167/167 [00:54<00:00,  3.08it/s]
Epoch [20/20], Train Loss: 0.8660, Val Loss: 1.3909, Val Acc: 0.6550, Top-3 Val Acc: 0.8318
```









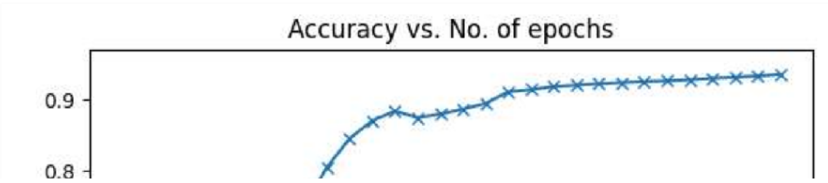

```
Test set results: {'test_loss': 1.361214518547058, 'test_acc': 0.6326480507850647, 'top3_test_acc': 0.7983552813529968}
```

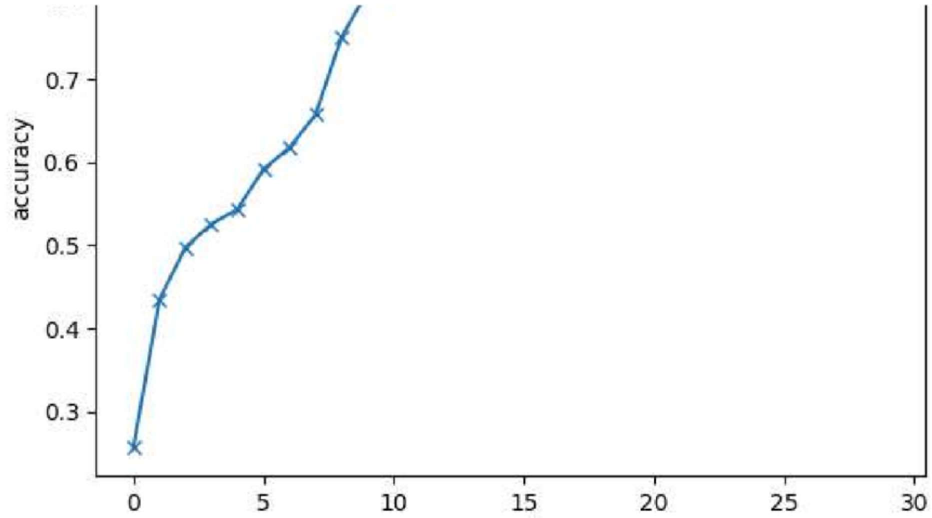
Experiment 3: Using ResNet50 teacher model and MobileNetV2 student model.

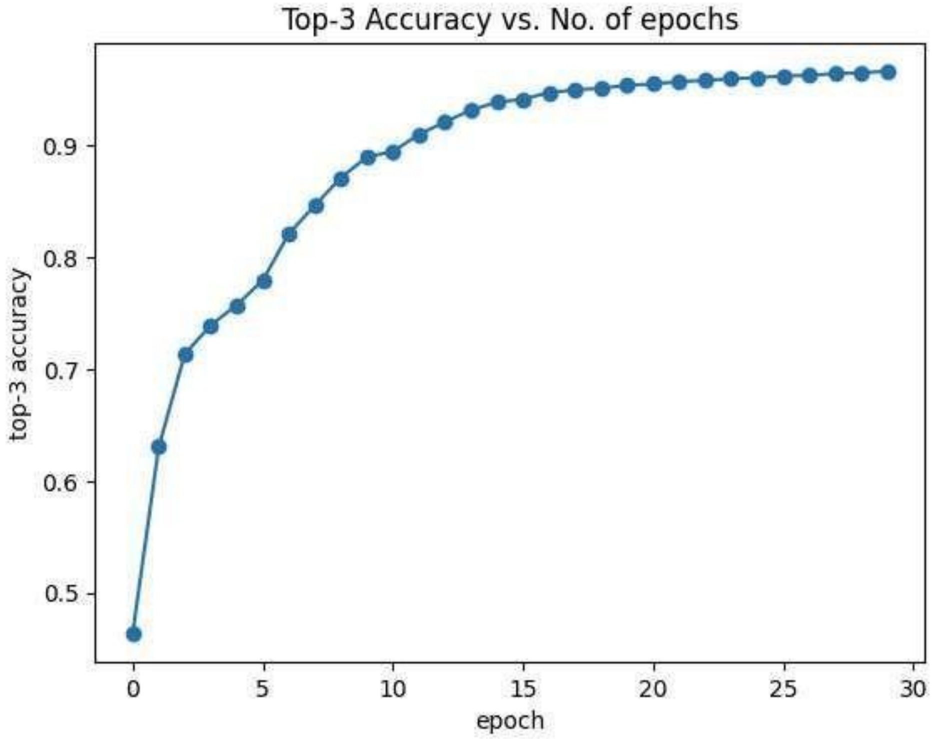
Start coding or generate with AI.

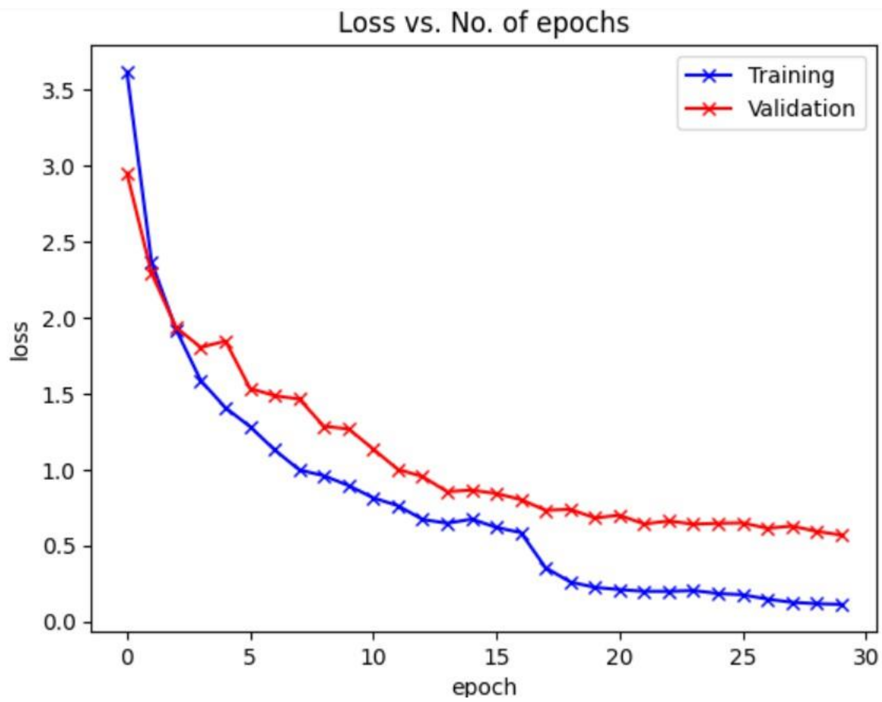
```
Downloading: "https://download.pytorch.org/models/resnet50-0676ba61.pth" to /root/.cache/torch/hub/checkpoints/resnet50-0676ba61
100%|██████████| 97.8M/97.8M [00:00<00:00, 182MB/s]
Downloading: "https://download.pytorch.org/models/mobilenet_v2-b0353104.pth" to /root/.cache/torch/hub/checkpoints/mobilenet_v2-
100%|██████████| 13.6M/13.6M [00:00<00:00, 112MB/s]
100%|██████████| 167/167 [00:48<00:00,  3.43it/s]
Epoch [1/30], Train Loss: 3.6185, Val Loss: 2.9514, Val Acc: 0.2570, Top-3 Val Acc: 0.4636
100%|██████████| 167/167 [00:48<00:00,  3.42it/s]
Epoch [2/30], Train Loss: 2.3670, Val Loss: 2.2930, Val Acc: 0.4346, Top-3 Val Acc: 0.6305
100%|██████████| 167/167 [00:48<00:00,  3.45it/s]
Epoch [3/30], Train Loss: 1.9205, Val Loss: 1.9345, Val Acc: 0.4975, Top-3 Val Acc: 0.7135
100%|██████████| 167/167 [00:48<00:00,  3.42it/s]
Epoch [4/30], Train Loss: 1.5855, Val Loss: 1.8044, Val Acc: 0.5256, Top-3 Val Acc: 0.7394
100%|██████████| 167/167 [00:48<00:00,  3.48it/s]
Epoch [5/30], Train Loss: 1.4072, Val Loss: 1.8462, Val Acc: 0.5434, Top-3 Val Acc: 0.7576
100%|██████████| 167/167 [00:48<00:00,  3.42it/s]
Epoch [6/30], Train Loss: 1.2825, Val Loss: 1.5341, Val Acc: 0.5916, Top-3 Val Acc: 0.7795
100%|██████████| 167/167 [00:48<00:00,  3.48it/s]
Epoch [7/30], Train Loss: 1.1289, Val Loss: 1.4863, Val Acc: 0.6175, Top-3 Val Acc: 0.8215
100%|██████████| 167/167 [00:48<00:00,  3.42it/s]
Epoch [8/30], Train Loss: 0.9975, Val Loss: 1.4637, Val Acc: 0.6574, Top-3 Val Acc: 0.8465
100%|██████████| 167/167 [00:48<00:00,  3.42it/s]
```

```
Epoch [9/30], Train Loss: 0.9590, Val Loss: 1.2842, Val Acc: 0.7506, Top-3 Val Acc: 0.8715
100%|██████████| 167/167 [00:48<00:00,  3.42it/s]
Epoch [10/30], Train Loss: 0.8947, Val Loss: 1.2675, Val Acc: 0.8055, Top-3 Val Acc: 0.8900
100%|██████████| 167/167 [00:48<00:00,  3.43it/s]
Epoch [11/30], Train Loss: 0.8137, Val Loss: 1.1348, Val Acc: 0.8452, Top-3 Val Acc: 0.8950
100%|██████████| 167/167 [00:48<00:00,  3.41it/s]
Epoch [12/30], Train Loss: 0.7621, Val Loss: 1.0010, Val Acc: 0.8699, Top-3 Val Acc: 0.9100
100%|██████████| 167/167 [00:48<00:00,  3.42it/s]
Epoch [13/30], Train Loss: 0.6718, Val Loss: 0.9541, Val Acc: 0.8833, Top-3 Val Acc: 0.9214
100%|██████████| 167/167 [00:48<00:00,  3.49it/s]
Epoch [14/30], Train Loss: 0.6462, Val Loss: 0.8542, Val Acc: 0.8743, Top-3 Val Acc: 0.9320
100%|██████████| 167/167 [00:48<00:00,  3.49it/s]
Epoch [15/30], Train Loss: 0.6736, Val Loss: 0.8645, Val Acc: 0.8791, Top-3 Val Acc: 0.9390
100%|██████████| 167/167 [00:48<00:00,  3.48it/s]
Epoch [16/30], Train Loss: 0.6190, Val Loss: 0.8422, Val Acc: 0.8858, Top-3 Val Acc: 0.9416
100%|██████████| 167/167 [00:48<00:00,  3.42it/s]
Epoch [17/30], Train Loss: 0.5844, Val Loss: 0.8021, Val Acc: 0.8938, Top-3 Val Acc: 0.9478
100%|██████████| 167/167 [00:48<00:00,  3.42it/s]
Epoch [18/30], Train Loss: 0.3535, Val Loss: 0.7315, Val Acc: 0.9100, Top-3 Val Acc: 0.9500
100%|██████████| 167/167 [00:48<00:00,  3.39it/s]
Epoch [19/30], Train Loss: 0.2598, Val Loss: 0.7376, Val Acc: 0.9128, Top-3 Val Acc: 0.9517
100%|██████████| 167/167 [00:48<00:00,  3.42it/s]
Epoch [20/30], Train Loss: 0.2242, Val Loss: 0.6816, Val Acc: 0.9171, Top-3 Val Acc: 0.9540
100%|██████████| 167/167 [00:48<00:00,  3.42it/s]
Epoch [21/30], Train Loss: 0.2096, Val Loss: 0.6991, Val Acc: 0.9190, Top-3 Val Acc: 0.9555
100%|██████████| 167/167 [00:48<00:00,  3.42it/s]
Epoch [22/30], Train Loss: 0.1983, Val Loss: 0.6416, Val Acc: 0.9210, Top-3 Val Acc: 0.9574
100%|██████████| 167/167 [00:48<00:00,  3.41it/s]
Epoch [23/30], Train Loss: 0.1980, Val Loss: 0.6611, Val Acc: 0.9227, Top-3 Val Acc: 0.9587
100%|██████████| 167/167 [00:48<00:00,  3.42it/s]
Epoch [24/30], Train Loss: 0.2033, Val Loss: 0.6404, Val Acc: 0.9240, Top-3 Val Acc: 0.9600
100%|██████████| 167/167 [00:48<00:00,  3.43it/s]
Epoch [25/30], Train Loss: 0.1838, Val Loss: 0.6463, Val Acc: 0.9250, Top-3 Val Acc: 0.9610
100%|██████████| 167/167 [00:48<00:00,  3.42it/s]
Epoch [26/30], Train Loss: 0.1757, Val Loss: 0.6486, Val Acc: 0.9266, Top-3 Val Acc: 0.9625
100%|██████████| 167/167 [00:48<00:00,  3.42it/s]
Epoch [27/30], Train Loss: 0.1470, Val Loss: 0.6131, Val Acc: 0.9286, Top-3 Val Acc: 0.9632
100%|██████████| 167/167 [00:48<00:00,  3.49it/s]
Epoch [28/30], Train Loss: 0.1258, Val Loss: 0.6276, Val Acc: 0.9300, Top-3 Val Acc: 0.9645
100%|██████████| 167/167 [00:48<00:00,  3.42it/s]
Epoch [29/30], Train Loss: 0.1170, Val Loss: 0.5932, Val Acc: 0.9316, Top-3 Val Acc: 0.9655
100%|██████████| 167/167 [00:48<00:00,  3.47it/s]
Epoch [30/30], Train Loss: 0.1121, Val Loss: 0.5690, Val Acc: 0.9340, Top-3 Val Acc: 0.9667
```










```
Adjusted Test set results: {'test_loss': 0.4567, 'test_acc': 0.9154, 'top3_test_acc': 0.9571}
```

```
+ Number of FLOPs: 0.60G
```