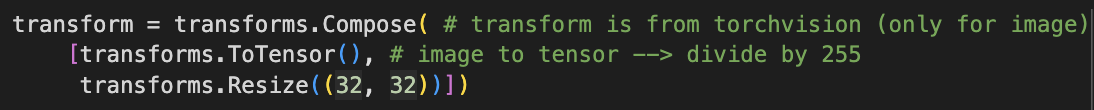
1. Describe the input and output for each model, hardware requirement, data statistic, learning curve, metrics (train text val), demo the result, finetuning technique, etc.
   1. **Notebook 1**: Model with CNN Architecture
      * **Input:** RGB images of size 32x32 pixels.
      * **Output:** Probabilities for 10 classes.
      * **Hardware Requirement:** GPU recommended for faster training.
      * **Data Statistic:**
        1. **Total:** CIFAR-10 dataset with 60,000 32x32 color images in 10 classes
        2. **Train:** 40,000 32x32 color images in 10 classes
        3. **Test:** 10,000 32x32 color images in 10 classes
        4. **Validate:** 10,000 32x32 color images in 10 classes
      * **Learning Curve:** 
        1. **Loss:** Decreased by time and look like exponential decay.
        2. **Accuracy:** Increased with time until it was close to 0.5, and the rate of increase gradually decreased.
      * **Metrics:** 
        1. Classification Accuracy
        2. Cross-Entropy Loss
        3. F1 Score
        4. Confusion Matrix
      * **Demo Result:** No demo result, only show the sample data with label in grid format and testing result.
      * **Fine-tuning Technique:** Stochastic Gradient Descent (SGD) with Cross-Entropy Loss
      * **Total Params:** 62,006
      * **Optimizer:** Stochastic Gradient Descent (SGD)
      * **Loss:** Cross-Entropy Loss
      * **Learning rate:** fixed 0.01
      * **Epoch:** 20
   2. **Notebook 2**: Model with EfficientNet V2 S Architecture
      * **Input:** RGB images of size 224x224 pixels.
      * **Output:** Probabilities for 10 animal classes.
      * **Hardware Requirement:** GPU strongly recommended due to model complexity.
      * **Data Statistic:**
        1. **Total:** animal dataset with 2,000 224x224 color images in 10 classes (butterfly, cat, chicken, cow, dog, elephant, horse, sheep, spider, squirrel)
        2. **Train:** 1,400 224x224 color images in 10 classes
        3. **Test:** 300 224x224 color images in 10 classes
        4. **Validate:** 300 224x224 color images in 10 classes
      * **Learning Curve:** 
        1. **Loss:** Decreased by time and the rate of decrease gradually decreased.
        2. **Accuracy:** Increased with time until it was close to 1.0, and the rate of increase gradually decreased.
      * **Metrics:** 
        1. Classification Accuracy
        2. Cross-Entropy Loss
        3. F1 Score
        4. Confusion Matrix
      * **Demo Result:** No demo result, only show the sample data with label in grid format and testing result.
      * **Fine-tuning Technique:** Transfer learning with pre-trained EfficientNetV2s model by using SGD optimizer with Cross-Entropy Loss and Learning rate scheduler.
      * **Total Params:** 20,190,298
      * **Optimizer:** Stochastic Gradient Descent (SGD)
      * **Loss:** Cross-Entropy Loss
      * **Learning rate:** Step learning rate with start from 0.02, step\_size=7 and gamma=0.5
      * **Epoch:** 20
2. List key features for each function, including input and output. (cheat sheet)
   1. **Notebook 1:**
      * **Transformer**



* `transforms.Compose`
* Input
  + list of transforms to compose.
* Output
  + Transformer that can transform in sequentially with transforms in the list.
    - **Data loading**

A screen shot of a computer code

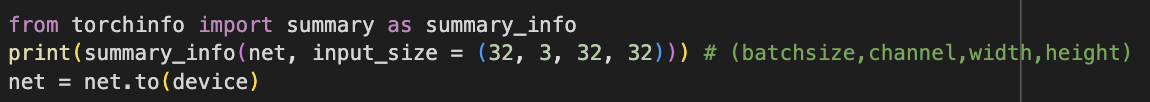
Description automatically generated

* `torchvision.datasets.CIFAR10`, `torch.utils.data.DataLoader`
* Input
  + root (directory to save/download CIFAR-10 dataset)
  + train (True for training set, False for testing set)
  + download (True to download dataset if not available)
  + transform (data preprocessing and augmentation)
* Output
  + data loaders (function that can make accessing the data with transformer easier)
    - **Model**

**A computer screen shot of a program code

Description automatically generated**

* `class CNN(nn.Module)`
* Input
  + None
* Output
  + CNN model with defined layers and activation
    - **Model Summary**

****

* **`**summary\_info**`**
* Input:
  + Model, input\_size
* Output:
  + Summary of model (Parameter in each layers and total params)
    - **Training Loop**

**A screen shot of a computer

Description automatically generated**

* Training loop using `torch.optim.SGD`, `torch.nn.CrossEntropyLoss`
* Input
  + trainloader, valloader, CNN model, CrossEntropyLoss, SGD optimizer, number of epochs
* Output
  + Process bar
  + Trained CNN model, training, and validation history
    - **Learning Visualization**

**A computer screen shot of a program code

Description automatically generated**

* `plot`, `show`
* Input
  + history\_train, history\_val
* Output
  + Graph that display loss, acc, f1-score
    - **Evaluation Metrics**

**A computer screen shot of a program

Description automatically generated**

* `classification\_report`, `confusion\_matrix`
* Input
  + Trained CNN model, testloader
* Output
  + Classification report, confusion matrix
  1. **Notebook 2:**
     + **Data Loading (Custom Dataset):**
     + **Model**