Name: Abrar Ahmed Mohammed

CARZAM Project Notebook

Downloading

```
import os
if not os.path.exists('./original_tool_image.zip'):
  ! wget -O original_tool_image.zip https://www.dropbox.com/s/pha9yzdfkmzoqob/original_tool_images.zip?dl=0
```

```
--2022-12-03 12:31:45--  https://www.dropbox.com/s/pha9yzdfkmzoqob/original_tool_images.zip?dl=0
Resolving www.dropbox.com (www.dropbox.com)... 162.125.3.18, 2620:100:601b:18::a27d:812
Connecting to www.dropbox.com (www.dropbox.com)|162.125.3.18|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: /s/raw/pha9yzdfkmzoqob/original_tool_images.zip [following]
--2022-12-03 12:31:45--  https://www.dropbox.com/s/raw/pha9yzdfkmzoqob/original_tool_images.zip
Reusing existing connection to www.dropbox.com:443.
HTTP request sent, awaiting response... 302 Found
Location: https://uc20617f0841f7dd35728328c931.dl.dropboxusercontent.com/cd/0/inline/Bx4KNe7vu4y38PmGoHX2zZNq8-r4RJkZ_-tfzKKxBmTJoz
--2022-12-03 12:31:45--  https://uc20617f0841f7dd35728328c931.dl.dropboxusercontent.com/cd/0/inline/Bx4KNe7vu4y38PmGoHX2zZNq8-r4RJk
Resolving uc20617f0841f7dd35728328c931.dl.dropboxusercontent.com (uc20617f0841f7dd35728328c931.dl.dropboxusercontent.com)... 162.12
Connecting to uc20617f0841f7dd35728328c931.dl.dropboxusercontent.com (uc20617f0841f7dd35728328c931.dl.dropboxusercontent.com)|162.1
HTTP request sent, awaiting response... 302 Found
Location: /cd/0/inline2/Bx4LzwzbdyHY88rwho_tr-HclZLQmx8TfS43Kjj9VMvAeLAE2jUbSytf12oAJQoc9up-E7EWO6hHziOpc0Za8qXRGPcqAvE861PucKeWAPM
--2022-12-03 12:31:46--  https://uc20617f0841f7dd35728328c931.dl.dropboxusercontent.com/cd/0/inline2/Bx4LzwzbdyHY88rwho_tr-HclZLQmx
Reusing existing connection to uc20617f0841f7dd35728328c931.dl.dropboxusercontent.com:443.
HTTP request sent, awaiting response... 200 OK
Length: 87282712 (83M) [application/zip]
Saving to: ‘original_tool_image.zip’

original_tool_image 100%[===================>]  83.24M  99.3MB/s    in 0.8s

2022-12-03 12:31:48 (99.3 MB/s) - ‘original_tool_image.zip’ saved [87282712/87282712]
```

Setup

Git Clone

From Source

```
! rm -rf -- GLAMOR
```

```
! git clone -b master https://github.com/asuprem/GLAMOR
```

```
Cloning into 'GLAMOR'...
remote: Enumerating objects: 8567, done.
remote: Counting objects: 100% (470/470), done.
remote: Compressing objects: 100% (296/296), done.
remote: Total 8567 (delta 247), reused 286 (delta 131), pack-reused 8097
Receiving objects: 100% (8567/8567), 2.34 MiB | 4.40 MiB/s, done.
Resolving deltas: 100% (5638/5638), done.
```

```
!pip install -e GLAMOR/
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Obtaining file:///content/GLAMOR
Requirement already satisfied: scikit-learn>=1.0.2 in /usr/local/lib/python3.8/dist-packages (from ednaml==0.1.5) (1.0.2)
Requirement already satisfied: torch>=1.10.* in /usr/local/lib/python3.8/dist-packages (from ednaml==0.1.5) (1.12.1+cu113)
Collecting torchinfo>=1.6.5
  Downloading torchinfo-1.7.1-py3-none-any.whl (22 kB)
Requirement already satisfied: torchvision>=0.11.* in /usr/local/lib/python3.8/dist-packages (from ednaml==0.1.5) (0.13.1+cu113)
Requirement already satisfied: Pillow>=7.1.2 in /usr/local/lib/python3.8/dist-packages (from ednaml==0.1.5) (7.1.2)
Requirement already satisfied: tqdm>=4.63.* in /usr/local/lib/python3.8/dist-packages (from ednaml==0.1.5) (4.64.1)
Collecting sentencepiece>=0.1.96
  Downloading sentencepiece-0.1.97-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (1.3 MB)
     |████████████████████████████████| 1.3 MB 7.8 MB/s
Requirement already satisfied: sortedcontainers>=2.4.0 in /usr/local/lib/python3.8/dist-packages (from ednaml==0.1.5) (2.4.0)
Requirement already satisfied: pyyaml>=6.0 in /usr/local/lib/python3.8/dist-packages (from ednaml==0.1.5) (6.0)
Requirement already satisfied: numpy>=1.14.6 in /usr/local/lib/python3.8/dist-packages (from scikit-learn>=1.0.2->ednaml==0.1.5) (1
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.8/dist-packages (from scikit-learn>=1.0.2->ednaml==0.
```

```
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.8/dist-packages (from scikit-learn>=1.0.2->ednaml==0.1.5) (1.
Requirement already satisfied: scipy>=1.1.0 in /usr/local/lib/python3.8/dist-packages (from scikit-learn>=1.0.2->ednaml==0.1.5) (1.
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.8/dist-packages (from torch>=1.10.*->ednaml==0.1.5) (4.1
Requirement already satisfied: requests in /usr/local/lib/python3.8/dist-packages (from torchvision>=0.11.*->ednaml==0.1.5) (2.23.0
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.8/dist-packages (from requests->torchvision>=0.11.*->ednaml==
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.8/dist-packages (from requests->torchvision>=0.11.*->edn
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.8/dist-packages (from requests->torchvision>=0.11.*->ed
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.8/dist-packages (from requests->to
Installing collected packages: torchinfo, sentencepiece, ednaml
  Running setup.py develop for ednaml
Successfully installed ednaml-0.1.5 sentencepiece-0.1.97 torchinfo-1.7.1
```

From PyPi

```
#! python -V
```

```
#! pip3 install --pre ednaml==0.1.4
```

Restart Notebook to Finish EdnaML Installation

Setting up

```
%load_ext autoreload
%autoreload 2
```

```
import torch
import ednaml
import glob, os
#from ednaml.core import EdnaDeploy, EdnaML
torch.__version__
```

```
'1.12.1+cu113'
```

Definitions: Crawler

```
# Here we define our custom model class
from ednaml.crawlers import Crawler
from zipfile import ZipFile # might be useful in unzipping!

class CarZamCrawler(Crawler):
  def __init__(self, logger, file_name = "original_tool_images.zip", **kwargs): # Add your own arguments if needed!
    self.classes = {}
    self.metadata = {}
    self.metadata["train"] = {}
    self.metadata["test"] = {}
    self.metadata["val"] = {}
    self.metadata["train"]["crawl"] = []  # <------ THIS NEEDS TO BE POPULATED
    self.metadata["test"]["crawl"] = []   # <------ THIS NEEDS TO BE POPULATED
    self.metadata["val"]["crawl"] = []    # <------ THIS NEEDS TO BE POPULATED

    # YOUR CODE HERE ------ POPULATE self.classes and self.metadata's empty lists ---
    from zipfile import ZipFile
    file_name = "original_tool_image.zip"
    fdest= "unzipped"
    if not os.path.exists(fdest):
      with ZipFile(file_name, 'r') as zip:
          # extract all files to another directory
          zip.extractall(fdest)
    fllist = glob.glob(os.path.join(fdest, "original_tool_images/*.jpg"))
    #tuple_prelim = [self.getinittuple(item) for item in fllist]
    temp=[]
    #ans=[]
    tokeep = ["Convertible", "Coupe", "Crossover", "Diesel", "Hybrid", "Sedan", "SUV", "Wagon","SportsCar", "Truck", "Van", ]
    tuple_expanded=[]
    for item in fllist:
      tuple_prelim=os.path.splitext(os.path.basename(item))[0].split(" "), item
      #print(tuple_prelim)
      if(len(tuple_prelim[0])==5):
        temp.append(tuple_prelim[1]) #appending path
        for i in tuple_prelim[0]:  #appending type,color,year,make,model
```

```
            if(i=="CoupeBlack"):
             temp.append("Coupe")
             temp.append("Black")
             temp.append(tuple_prelim[1:])
            else:
              temp.append(i)

        my_tuple=tuple(temp)        #list to Tuple
        tuple_expanded.append(my_tuple)
        temp=[]
      elif(len(tuple_prelim[0])==6):
        temp.append(tuple_prelim[1])
        if(tuple_prelim[0][0]) in tokeep:
          for i in tuple_prelim[0][:4]:
            temp.append(i)
          temp.append(tuple_prelim[0][4]+tuple_prelim[0][5])

          my_tuple=tuple(temp)
          tuple_expanded.append(my_tuple)
          temp=[]
        else:
          temp.append(tuple_prelim[0][0]+tuple_prelim[0][1])
          for i in tuple_prelim[0][2:]:
            temp.append(i)
          #temp.append(tuple_prelim[1])
          my_tuple=tuple(temp)
          tuple_expanded.append(my_tuple)
          temp=[]
#print(ans)
      elif(len(tuple_prelim[0])==7):
        temp.append(tuple_prelim[1])
        if tuple_prelim[0][0] in tokeep:
          for i in tuple_prelim[0][:4]:
            temp.append(i)
          temp.append(tuple_prelim[0][4:])

          my_tuple=tuple(temp)
          tuple_expanded.append(my_tuple)
          temp=[]
        else:
          temp.append(tuple_prelim[0][0]+tuple_prelim[0][1])
          for i in tuple_prelim[0][2:5]:
            temp.append(i)
          temp.append(tuple_prelim[0][5]+tuple_prelim[0][6])

          my_tuple=tuple(temp)
          tuple_expanded.append(my_tuple)
          temp=[]
    #print("Tuple_expanded:")
    #print(tuple_expanded[0])
    import random
    random.seed(3456)
    random.shuffle(tuple_expanded)

    splits = 0.8
    train_sets = int(len(tuple_expanded)*0.8)
    val_sets = int(len(tuple_expanded)*0.1)



    # structure:  (path, type, color, year, make)
    # idx           0     1     2     3     4
    print("Outliers\n")
    for item in tuple_expanded:
      if(item[1]=='CoupeBlack'):
        print(item)
        print("\n")

    types = list(set([item[1] for item in tuple_expanded]))
    print("types:",types)
    colors = list(set([item[2] for item in tuple_expanded]))
    years = list(set([item[3] for item in tuple_expanded]))
    makes = list(set([item[4] for item in tuple_expanded]))

    self.classes["vtype"] = len(types)
    self.classes["color"] = len(colors)
    self.classes["year"] = len(years)
    self.classes["make"] = len(makes)

    self.type_lookup = {item:idx for idx,item in enumerate(types)}
    self.color_lookup = {item:idx for idx,item in enumerate(colors)}
    self.year_lookup = {item:idx for idx,item in enumerate(years)}
```

```
    self.make_lookup = {item:idx for idx,item in enumerate(makes)}

    tuple_ex=tuple_expanded
    tuple_expanded = [(item[0], self.type_lookup[item[1]], self.color_lookup[item[2]], self.year_lookup[item[3]], self.make_lookup[item[4
    print("\n")
    print("Tuple_expanded\n")
    print(tuple_expanded[:5])

    #split the datasets
    self.metadata["train"]["crawl"] = tuple_expanded[:train_sets]
    self.metadata["val"]["crawl"] = tuple_expanded[train_sets:val_sets]
    self.metadata["test"]["crawl"] = tuple_expanded[train_sets+val_sets:]


    # ----------------------------------------------------------------------------

    self.metadata["train"]["classes"] = self.classes
    self.metadata["test"]["classes"] = self.classes
    self.metadata["val"]["classes"] = self.classes

  def getinittuple(self, item):
    return (os.path.splitext(os.path.basename(item))[0].split(" "), item)
```

Testing the Crawler

```
kwargs = {
    "logger" : None,
    "file_name" : "original_tool_images.zip",
    # add any other kwargs here...
}
```

```
crawler = CarZamCrawler(**kwargs)
```

```
Outliers

types: ['Coupe', 'Wagon', 'Hybrid', 'LuxuryVehicle', 'Diesel', 'SUV', 'Convertible', 'PickupTruck', 'Van', 'Crossover', 'ElectricVe
Tuple_expanded

[('unzipped/original_tool_images/Sedan Blue 2016 Mercedes-Benz E350.jpg', 12, 0, 12, 19), ('unzipped/original_tool_images/Crossover
```

```
crawler.classes # You should get the classes here
```

```
{'vtype': 13, 'color': 13, 'year': 14, 'make': 36}
```

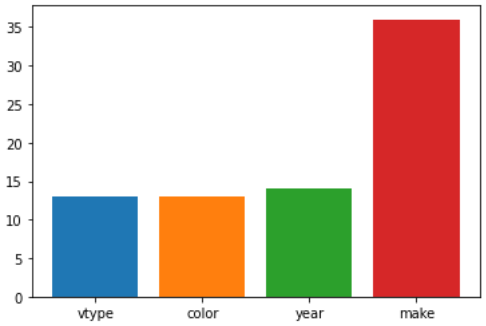
```
crawler.metadata["train"]["crawl"][:5]  # You should get the list of tuples here
```

```
[('unzipped/original_tool_images/Sedan Blue 2016 Mercedes-Benz E350.jpg',
  12,
  0,
  12,
  19),
 ('unzipped/original_tool_images/Crossover White 2017 Chevrolet Equinox.jpg',
  9,
  9,
  4,
  23),
 ('unzipped/original_tool_images/Sports Car White 2017 Chevrolet Corvette.jpg',
  11,
  9,
  4,
  23),
 ('unzipped/original_tool_images/Sedan Silver 2020 Honda Accord.jpg',
  12,
  11,
  8,
  27),
 ('unzipped/original_tool_images/Sedan Black 2015 BMW 550.jpg', 12, 7, 10, 6)]
```

Statistics

```
import matplotlib.pyplot as plt
```

```
# Write code to collect info on # makes, models, year, type
#
#
for i in crawler.classes:
  plt.bar(i,crawler.classes[i])
```



3.1 Single classification (Vehicle Type)

```
class_name = "vtype"  # Make sure to change this to whatever name you used for make in your `original_tool_images` crawler
class_idx = 1         # Make sure to change this to whetever index `type` is in your Crawler's tuple!
path_idx = 0          # Change this to whichever index in tuple has path
crawler_args = {"file_name" : "original_tool_image.zip"}
```

```
%load_ext autoreload
%autoreload 2
```

```
     The autoreload extension is already loaded. To reload it, use:
       %reload_ext autoreload
```

```
from ednaml.core import EdnaML
from ednaml.generators import ClassificationGenerator

eml = EdnaML(config = "./GLAMOR/profiles/CarZam/base_config.yml", config_inject=[
    ("SAVE.MODEL_QUALIFIER", class_name)
])

eml.cfg.EXECUTION.DATAREADER.CRAWLER_ARGS = crawler_args
eml.cfg.EXECUTION.DATAREADER.DATASET_ARGS["pathidx"] = path_idx
eml.cfg.EXECUTION.DATAREADER.DATASET_ARGS["annotationidx"] = class_idx
eml.cfg.EXECUTION.DATAREADER.DATASET_ARGS["classificationclass"] = class_name

eml.addGeneratorClass(ClassificationGenerator)
eml.addCrawlerClass(CarZamCrawler)
```

```
     Injected key-value pair:  SAVE.MODEL_QUALIFIER, vtype
```

```
eml.apply()
```

```
03:26:57 Running classification model with classes: {'vtype': {'classes': 13}}
03:26:57 Generated test data/query generator
03:26:57 Loaded classification_model_builder from ednaml.models to build model
Outliers

types: ['ElectricVehicle', 'Wagon', 'Coupe', 'SUV', 'Van', 'PickupTruck', 'Sedan', 'Hybrid', 'LuxuryVehicle', 'Convertible', 'Di
Tuple_expanded

[('unzipped/original_tool_images/Hybrid Silver 2021 Toyota Avalon Hybrid.jpg', 7, 2, 0, 7), ('unzipped/original_tool_images/Cros
03:26:58 Finished instantiating model with ClassificationResnet architecture
03:26:58 Adding plugins after constructing model
03:26:58 No saved model weights provided.
03:27:02 Model Summary retured the following error:
03:27:02 Traceback (most recent call last):
  File "/content/GLAMOR/src/ednaml/core/EdnaML.py", line 888, in getModelSummary
    self.cfg.TRAIN_TRANSFORMATION.INPUT_SIZE,
AttributeError: 'TransformationConfig' object has no attribute 'INPUT_SIZE'

03:27:02 Loaded ClassificationOptimizer from ednaml.optimizer to build Optimizer model
03:27:02 Built optimizer
03:27:02 Built scheduler
03:27:02 Added SoftmaxLogitsLoss with lambda = 1.0 and loss arguments {}
03:27:02 Built loss function
03:27:02 Built loss optimizer
03:27:02 Built loss scheduler
03:27:02 Loaded BaseStorage from ednaml.storage to build Storage
03:27:02 Loaded ClassificationTrainer from ednaml.trainer to build Trainer
03:27:02 Saving model metadata
03:27:02 Backing up metadata
```

```
eml.train()
```

```
03:30:12 Final: Saving model at save-frequency
03:30:12 Saving model, optimizer, and scheduler.
```

```
resp = eml.eval()
```

```
03:30:32 Obtained features, validation in progress
03:30:32 Accuracy: 41.414%
03:30:32 Micro F-score: 0.414
03:30:32 Weighted F-score: 0.321
```

3.2 Single classification (Vehicle Color)

```
class_name = "color"   # Make sure to change this to whatever name you used for make in your `original_tool_images` crawler
class_idx = 2          # Make sure to change this to whetever index `color` is in your Crawler's tuple!
path_idx = 0           # Change this to whichever index in tuple has path
crawler_args = {"file_name" : "original_tool_image.zip"}
```

```
%load_ext autoreload
%autoreload 2
```

```
The autoreload extension is already loaded. To reload it, use:
  %reload_ext autoreload
```

```
from ednaml.core import EdnaML
from ednaml.generators import ClassificationGenerator

eml = EdnaML(config = "./GLAMOR/profiles/CarZam/base_config.yml", config_inject=[
    ("SAVE.MODEL_QUALIFIER", class_name)
])

eml.cfg.EXECUTION.DATAREADER.CRAWLER_ARGS = crawler_args
eml.cfg.EXECUTION.DATAREADER.DATASET_ARGS["pathidx"] = path_idx
eml.cfg.EXECUTION.DATAREADER.DATASET_ARGS["annotationidx"] = class_idx
eml.cfg.EXECUTION.DATAREADER.DATASET_ARGS["classificationclass"] = class_name

eml.addGeneratorClass(ClassificationGenerator)
eml.addCrawlerClass(CarZamCrawler)
```

```
Injected key-value pair:  SAVE.MODEL_QUALIFIER, color
```

```
eml.apply()
```

```
AttributeError: 'TransformationConfig' object has no attribute 'INPUT_SIZE'

03:35:25 Loaded ClassificationOptimizer from ednaml.optimizer to build Optimizer model
03:35:25 Built optimizer
03:35:25 Built scheduler
03:35:25 Added SoftmaxLogitsLoss with lambda = 1.0 and loss arguments {}
03:35:25 Built loss function
03:35:25 Built loss optimizer
03:35:25 Built loss scheduler
03:35:25 Loaded BaseStorage from ednaml.storage to build Storage
03:35:25 Loaded ClassificationTrainer from ednaml.trainer to build Trainer
03:35:25 Saving model metadata
03:35:25 Backing up metadata
03:35:25 Finished metadata backup
```

```
eml.train()
```

```
03:37:16 Micro F-score: 0.712
03:37:16 Weighted F-score: 0.659
03:37:16 Saving model at save-frequency, at epoch 5, step 0
03:37:16 Saving model, optimizer, and scheduler.
03:37:29 ********** Completed epoch 6 **********
03:37:29 Model evaluation triggered, but gradients still need accumulation. Will evaluate after accumulation.
03:37:29 Model save triggered, but gradients still need accumulation. Will save after accumulation.
03:37:29 Parameter Group `opt-1`: Starting epoch 7 with 50 steps and learning rate 1.00000E-05
03:37:29 Evaluating model at test-frequency
03:37:32 Obtained features, validation in progress
03:37:32 Accuracy: 72.222%
03:37:32 Micro F-score: 0.722
03:37:32 Weighted F-score: 0.676
03:37:32 Saving model at save-frequency, at epoch 6, step 0
03:37:32 Saving model, optimizer, and scheduler.
03:37:46 ********** Completed epoch 7 **********
03:37:46 Model evaluation triggered, but gradients still need accumulation. Will evaluate after accumulation.
03:37:46 Model save triggered, but gradients still need accumulation. Will save after accumulation.
03:37:47 Parameter Group `opt-1`: Starting epoch 8 with 50 steps and learning rate 1.00000E-05
03:37:47 Evaluating model at test-frequency
03:37:50 Obtained features, validation in progress
03:37:50 Accuracy: 72.727%
03:37:50 Micro F-score: 0.727
03:37:50 Weighted F-score: 0.679
03:37:50 Saving model at save-frequency, at epoch 7, step 0
03:37:50 Saving model, optimizer, and scheduler.
03:38:03 ********** Completed epoch 8 **********
03:38:03 Model evaluation triggered, but gradients still need accumulation. Will evaluate after accumulation.
03:38:03 Model save triggered, but gradients still need accumulation. Will save after accumulation.
03:38:03 Parameter Group `opt-1`: Starting epoch 9 with 50 steps and learning rate 1.00000E-05
03:38:04 Evaluating model at test-frequency
03:38:06 Obtained features, validation in progress
03:38:06 Accuracy: 74.747%
03:38:06 Micro F-score: 0.747
03:38:06 Weighted F-score: 0.706
03:38:06 Saving model at save-frequency, at epoch 8, step 0
03:38:06 Saving model, optimizer, and scheduler.
03:38:19 ********** Completed epoch 9 **********
03:38:19 Model evaluation triggered, but gradients still need accumulation. Will evaluate after accumulation.
03:38:19 Model save triggered, but gradients still need accumulation. Will save after accumulation.
03:38:20 Parameter Group `opt-1`: Starting epoch 10 with 50 steps and learning rate 1.00000E-05
03:38:21 Evaluating model at test-frequency
03:38:24 Obtained features, validation in progress
03:38:24 Accuracy: 75.253%
03:38:24 Micro F-score: 0.753
03:38:24 Weighted F-score: 0.709
03:38:24 Saving model at save-frequency, at epoch 9, step 0
03:38:24 Saving model, optimizer, and scheduler.
03:38:36 ********** Completed epoch 10 **********
03:38:36 Model evaluation triggered, but gradients still need accumulation. Will evaluate after accumulation.
03:38:36 Model save triggered, but gradients still need accumulation. Will save after accumulation.
03:38:36 Final: Evaluating model at test-frequency
03:38:38 Obtained features, validation in progress
03:38:38 Accuracy: 76.768%
03:38:38 Micro F-score: 0.768
03:38:38 Weighted F-score: 0.727
03:38:38 Final: Saving model at save-frequency
03:38:38 Saving model, optimizer, and scheduler.
```

```
resp = eml.eval()
```

```
03:39:17 Obtained features, validation in progress
03:39:17 Accuracy: 76.768%
03:39:17 Micro F-score: 0.768
03:39:17 Weighted F-score: 0.727
```

3.3 Single classification (Vehicle Make)

```
class_name = "make"   # Make sure to change this to whatever name you used for make in your `original_tool_images` crawler
class_idx = 4         # Make sure to change this to whetever index `make` is in your Crawler's tuple!
path_idx = 0          # Change this to whichever index in tuple has path
crawler_args = {"file_name" : "original_tool_image.zip"}


%load_ext autoreload
%autoreload 2
```

```
     The autoreload extension is already loaded. To reload it, use:
       %reload_ext autoreload
```

```
from ednaml.core import EdnaML
from ednaml.generators import ClassificationGenerator

eml = EdnaML(config = "./GLAMOR/profiles/CarZam/base_config.yml", config_inject=[
    ("SAVE.MODEL_QUALIFIER", class_name)
])

eml.cfg.EXECUTION.DATAREADER.CRAWLER_ARGS = crawler_args
eml.cfg.EXECUTION.DATAREADER.DATASET_ARGS["pathidx"] = path_idx
eml.cfg.EXECUTION.DATAREADER.DATASET_ARGS["annotationidx"] = class_idx
eml.cfg.EXECUTION.DATAREADER.DATASET_ARGS["classificationclass"] = class_name

eml.addGeneratorClass(ClassificationGenerator)
eml.addCrawlerClass(CarZamCrawler)
```

```
     Injected key-value pair:  SAVE.MODEL_QUALIFIER, make
```

```
eml.apply()
```

```
eml.train()
```

```
03:42:10 Obtained features, validation in progress
03:42:10 Accuracy: 41.919%
03:42:10 Micro F-score: 0.419
03:42:10 Weighted F-score: 0.401
03:42:10 Saving model at save-frequency, at epoch 3, step 0
03:42:10 Saving model, optimizer, and scheduler.
03:42:23 ********** Completed epoch 4 **********
03:42:23 Model evaluation triggered, but gradients still need accumulation. Will evaluate after accumulation.
03:42:23 Model save triggered, but gradients still need accumulation. Will save after accumulation.
03:42:23 Parameter Group `opt-1`: Starting epoch 5 with 50 steps and learning rate 1.00000E-05
03:42:23 Evaluating model at test-frequency
03:42:26 Obtained features, validation in progress
03:42:26 Accuracy: 47.475%
03:42:26 Micro F-score: 0.475
03:42:26 Weighted F-score: 0.461
03:42:26 Saving model at save-frequency, at epoch 4, step 0
03:42:26 Saving model, optimizer, and scheduler.
03:42:39 ********** Completed epoch 5 **********
03:42:39 Model evaluation triggered, but gradients still need accumulation. Will evaluate after accumulation.
03:42:39 Model save triggered, but gradients still need accumulation. Will save after accumulation.
03:42:39 Parameter Group `opt-1`: Starting epoch 6 with 50 steps and learning rate 1.00000E-05
03:42:40 Evaluating model at test-frequency
03:42:43 Obtained features, validation in progress
03:42:43 Accuracy: 49.495%
03:42:43 Micro F-score: 0.495
03:42:43 Weighted F-score: 0.468
03:42:43 Saving model at save-frequency, at epoch 5, step 0
03:42:43 Saving model, optimizer, and scheduler.
03:42:55 ********** Completed epoch 6 **********
03:42:55 Model evaluation triggered, but gradients still need accumulation. Will evaluate after accumulation.
03:42:55 Model save triggered, but gradients still need accumulation. Will save after accumulation.
03:42:55 Parameter Group `opt-1`: Starting epoch 7 with 50 steps and learning rate 1.00000E-05
03:42:55 Evaluating model at test-frequency
03:42:58 Obtained features, validation in progress
03:42:58 Accuracy: 52.525%
03:42:58 Micro F-score: 0.525
03:42:58 Weighted F-score: 0.496
03:42:58 Saving model at save-frequency, at epoch 6, step 0
03:42:58 Saving model, optimizer, and scheduler.
03:43:10 ********** Completed epoch 7 **********
03:43:10 Model evaluation triggered, but gradients still need accumulation. Will evaluate after accumulation.
03:43:10 Model save triggered, but gradients still need accumulation. Will save after accumulation.
03:43:11 Parameter Group `opt-1`: Starting epoch 8 with 50 steps and learning rate 1.00000E-05
03:43:12 Evaluating model at test-frequency
03:43:15 Obtained features, validation in progress
03:43:15 Accuracy: 53.535%
03:43:15 Micro F-score: 0.535
03:43:15 Weighted F-score: 0.501
03:43:15 Saving model at save-frequency, at epoch 7, step 0
03:43:15 Saving model, optimizer, and scheduler.
03:43:27 ********** Completed epoch 8 **********
03:43:27 Model evaluation triggered, but gradients still need accumulation. Will evaluate after accumulation.
03:43:27 Model save triggered, but gradients still need accumulation. Will save after accumulation.
03:43:27 Parameter Group `opt-1`: Starting epoch 9 with 50 steps and learning rate 1.00000E-05
03:43:28 Evaluating model at test-frequency
03:43:30 Obtained features, validation in progress
03:43:30 Accuracy: 55.051%
03:43:30 Micro F-score: 0.551
03:43:30 Weighted F-score: 0.512
```

```
resp = eml.eval()
```

```
03:44:19 Obtained features, validation in progress
03:44:19 Accuracy: 53.030%
03:44:19 Micro F-score: 0.530
03:44:19 Weighted F-score: 0.477
```

4. Multiclass classifiers Multiclass classifiers try to classify multiple things at once, using the same features. Sometimes it works, if the features are colocated or have some overlap. Othertimes, it doesn't work very well. We can examine this in case of our small dataset first.

4.1 Multi-class classification (color-type) Now we will try a model that performs vehicle type AND vehicle color classification together. The config is already prepared for this in profiles/color_type.yml.

```
path_idx = 0          # Change this to whichever index in tuple has path
crawler_args = {"file_name" : "original_tool_image.zip"}
```

```
%load_ext autoreload
%autoreload 2
```

```
The autoreload extension is already loaded. To reload it, use:
  %reload_ext autoreload
```

```
from ednaml.core import EdnaML
from ednaml.generators import ClassificationGenerator

eml = EdnaML(config = ["./GLAMOR/profiles/CarZam/base_config.yml","./GLAMOR/profiles/CarZam/color_type.yml"])

eml.cfg.EXECUTION.DATAREADER.CRAWLER_ARGS = crawler_args
eml.cfg.EXECUTION.DATAREADER.DATASET_ARGS["pathidx"] = path_idx
# We have already set these in config
#eml.cfg.EXECUTION.DATAREADER.DATASET_ARGS["annotationidx"] = class_idx
#eml.cfg.EXECUTION.DATAREADER.DATASET_ARGS["classificationclass"] = class_name

#eml.addGeneratorClass(MultiClassificationGenerator)
eml.addCrawlerClass(CarZamCrawler)
```

```
eml.apply()
```

```
- LOSS
- OPTIMIZER
- SCHEDULER
- LOSS_OPTIMIZER
- LOSS_SCHEDULER
- LOGGING
- DEPLOYMENT
- MODEL_PLUGIN

06:21:07
06:21:07
06:21:07 ****************************************
06:21:07 Model weights file resnet18-5c106cde.pth does not exist. Downloading.
46827520/46827520 bytes [                                                                                                    ]06
06:21:07 Loaded BaseStorage from ednaml.storage to build Storage
06:21:07 Reading data with DataReader DataReader
06:21:07 Default CRAWLER is <class 'ednaml.crawlers.Crawler'>

Download of resnet18-5c106cde.pth to https://download.pytorch.org/models/resnet18-5c106cde.pth completed
06:21:07 Default DATASET is <class 'torch.utils.data.dataset.Dataset'>
06:21:07 Default GENERATOR is <class 'ednaml.generators.ImageGenerator.ImageGenerator'>
06:21:07 Updating GENERATOR using config specification to MultiClassificationGenerator
06:21:07 Updating CRAWLER to CarZamCrawler
Outliers

types: ['Convertible', 'PickupTruck', 'Hybrid', 'Van', 'ElectricVehicle', 'Diesel', 'Crossover', 'Wagon', 'LuxuryVehicle', 'Seda
Tuple_expanded

[('unzipped/original_tool_images/Hybrid Silver 2021 Toyota Avalon Hybrid.jpg', 2, 0, 2, 22), ('unzipped/original_tool_images/Cro
06:21:07 Generated training data generator with 1575 training data points
06:21:07 Running classification model with classes: {'color': {'classes': 13}, 'vtype': {'classes': 13}}
06:21:07 Generated test data/query generator
06:21:07 Loaded multiclassification_model_builder from ednaml.models to build model
06:21:08 Finished instantiating model with MultiClassificationResnet architecture
06:21:08 Adding plugins after constructing model
06:21:08 No saved model weights provided.
06:21:12 Model Summary retured the following error:
06:21:12 Traceback (most recent call last):
  File "/content/GLAMOR/src/ednaml/core/EdnaML.py", line 888, in getModelSummary
    self.cfg.TRAIN_TRANSFORMATION.INPUT_SIZE,
AttributeError: 'TransformationConfig' object has no attribute 'INPUT_SIZE'

06:21:12 Loaded ClassificationOptimizer from ednaml.optimizer to build Optimizer model
06:21:12 Built optimizer
06:21:12 Built scheduler
06:21:12 Added SoftmaxLogitsLoss with lambda = 1.0 and loss arguments {}
06:21:12 Added SoftmaxLogitsLoss with lambda = 1.0 and loss arguments {}
06:21:12 Built loss function
06:21:12 Built loss optimizer
06:21:12 Built loss scheduler
06:21:12 Built loss scheduler
06:21:12 Loaded BaseStorage from ednaml.storage to build Storage
06:21:12 Loaded MultiClassificationTrainer from ednaml.trainer to build Trainer
06:21:12 Saving model metadata
06:21:12 Backing up metadata
06:21:12 Finished metadata backup
06:21:12 1 GPUs available
```

```
eml.train()
```

```
06:23:19 Accuracy       color: 0.722    vtype: 0.348
06:23:19 M F-Score      color: 0.722    vtype: 0.348
06:23:19 W F-Score      color: 0.681    vtype: 0.302
06:23:19 Saving model at save-frequency, at epoch 6, step 0
06:23:19 Saving model, optimizer, and scheduler.
06:23:32 ********** Completed epoch 7 **********
06:23:32 Model evaluation triggered, but gradients still need accumulation. Will evaluate after accumulation.
06:23:32 Model save triggered, but gradients still need accumulation. Will save after accumulation.
06:23:32 Parameter Group `opt-1`: Starting epoch 8 with 50 steps and learning rate 1.00000E-05
06:23:33 Evaluating model at test-frequency
06:23:36 Obtained features, validation in progress
06:23:36 Metrics        colorloss       typeloss
06:23:36 Accuracy       color: 0.732    vtype: 0.318
06:23:36 M F-Score      color: 0.732    vtype: 0.318
06:23:36 W F-Score      color: 0.689    vtype: 0.271
06:23:36 Saving model at save-frequency, at epoch 7, step 0
06:23:36 Saving model, optimizer, and scheduler.
06:23:47 ********** Completed epoch 8 **********
06:23:47 Model evaluation triggered, but gradients still need accumulation. Will evaluate after accumulation.
06:23:47 Model save triggered, but gradients still need accumulation. Will save after accumulation.
06:23:48 Parameter Group `opt-1`: Starting epoch 9 with 50 steps and learning rate 1.00000E-05
06:23:48 Evaluating model at test-frequency
06:23:51 Obtained features, validation in progress
06:23:51 Metrics        colorloss       typeloss
06:23:51 Accuracy       color: 0.732    vtype: 0.323
06:23:51 M F-Score      color: 0.732    vtype: 0.323
06:23:51 W F-Score      color: 0.690    vtype: 0.280
06:23:51 Saving model at save-frequency, at epoch 8, step 0
06:23:51 Saving model, optimizer, and scheduler.
06:24:03 ********** Completed epoch 9 **********
06:24:03 Model evaluation triggered, but gradients still need accumulation. Will evaluate after accumulation.
06:24:03 Model save triggered, but gradients still need accumulation. Will save after accumulation.
06:24:03 Parameter Group `opt-1`: Starting epoch 10 with 50 steps and learning rate 1.00000E-05
06:24:04 Evaluating model at test-frequency
06:24:07 Obtained features, validation in progress
06:24:07 Metrics        colorloss       typeloss
06:24:07 Accuracy       color: 0.737    vtype: 0.313
06:24:07 M F-Score      color: 0.737    vtype: 0.313
06:24:07 W F-Score      color: 0.696    vtype: 0.260
06:24:07 Saving model at save-frequency, at epoch 9, step 0
06:24:07 Saving model, optimizer, and scheduler.
06:24:19 ********** Completed epoch 10 **********
06:24:19 Model evaluation triggered, but gradients still need accumulation. Will evaluate after accumulation.
06:24:19 Model save triggered, but gradients still need accumulation. Will save after accumulation.
06:24:19 Final: Evaluating model at test-frequency
06:24:20 Obtained features, validation in progress
06:24:20 Metrics        colorloss       typeloss
06:24:20 Accuracy       color: 0.742    vtype: 0.313
06:24:20 M F-Score      color: 0.742    vtype: 0.313
06:24:20 W F-Score      color: 0.701    vtype: 0.259
06:24:20 Final: Saving model at save-frequency
06:24:20 Saving model, optimizer, and scheduler.
```

```
resp = eml.eval()
```

```
06:25:30 Obtained features, validation in progress
06:25:30 Metrics        colorloss       typeloss
06:25:30 Accuracy       color: 0.742    vtype: 0.313
06:25:30 M F-Score      color: 0.742    vtype: 0.313
06:25:30 W F-Score      color: 0.701    vtype: 0.259
```

4.2 Multi-class classification (color-type-make)

Now we will try a model that performs vehicle type vehicle color, and vehicle make classification together. The config is already prepared for this in profiles/color_type_make.yml

```
path_idx = 0          # Change this to whichever index in tuple has path
crawler_args = {"file_name" : "original_tool_image.zip"}
```

```
%load_ext autoreload
%autoreload 2
```

```
The autoreload extension is already loaded. To reload it, use:
  %reload_ext autoreload
```

```
from ednaml.core import EdnaML
from ednaml.generators import ClassificationGenerator

eml = EdnaML(config = ["./GLAMOR/profiles/CarZam/base_config.yml","./GLAMOR/profiles/CarZam/color_type_make.yml"])

eml.cfg.EXECUTION.DATAREADER.CRAWLER_ARGS = crawler_args
eml.cfg.EXECUTION.DATAREADER.DATASET_ARGS["pathidx"] = path_idx
# We have already set these in config
```

```
#eml.cfg.EXECUTION.DATAREADER.DATASET_ARGS["annotationidx"] = class_idx
#eml.cfg.EXECUTION.DATAREADER.DATASET_ARGS["classificationclass"] = class_name

#eml.addGeneratorClass(ClassificationGenerator)
eml.addCrawlerClass(CarZamCrawler)


eml.apply()
```

```
- MODEL
- LOSS
- OPTIMIZER
- SCHEDULER
- LOSS_OPTIMIZER
- LOSS_SCHEDULER
- LOGGING
- DEPLOYMENT
- MODEL_PLUGIN

06:31:14
06:31:14
06:31:14 ****************************************
06:31:14 No previous stop detected. Will start from epoch 0
06:31:14 Loaded BaseStorage from ednaml.storage to build Storage
06:31:14 Reading data with DataReader DataReader
06:31:14 Default CRAWLER is <class 'ednaml.crawlers.Crawler'>
06:31:14 Default DATASET is <class 'torch.utils.data.dataset.Dataset'>
06:31:14 Default GENERATOR is <class 'ednaml.generators.ImageGenerator.ImageGenerator'>
06:31:14 Updating GENERATOR using config specification to MultiClassificationGenerator
06:31:14 Updating CRAWLER to CarZamCrawler
06:31:14 Generated training data generator with 1575 training data points
06:31:14 Running classification model with classes: {'color': {'classes': 13}, 'vtype': {'classes': 13}, 'make': {'classes': 36}
06:31:14 Generated test data/query generator
06:31:14 Loaded multiclassification_model_builder from ednaml.models to build model
Outliers

types: ['Convertible', 'PickupTruck', 'Hybrid', 'Van', 'ElectricVehicle', 'Diesel', 'Crossover', 'Wagon', 'LuxuryVehicle', 'Seda
Tuple_expanded

[('unzipped/original_tool_images/Hybrid Silver 2021 Toyota Avalon Hybrid.jpg', 2, 0, 2, 22), ('unzipped/original_tool_images/Cro
06:31:14 Finished instantiating model with MultiClassificationResnet architecture
06:31:14 Adding plugins after constructing model
06:31:14 No saved model weights provided.
06:31:14 Model Summary retured the following error:
06:31:14 Traceback (most recent call last):
  File "/content/GLAMOR/src/ednaml/core/EdnaML.py", line 888, in getModelSummary
    self.cfg.TRAIN_TRANSFORMATION.INPUT_SIZE,
AttributeError: 'TransformationConfig' object has no attribute 'INPUT_SIZE'

06:31:14 Loaded ClassificationOptimizer from ednaml.optimizer to build Optimizer model
06:31:14 Built optimizer
06:31:14 Built scheduler
06:31:14 Added SoftmaxLogitsLoss with lambda = 1.0 and loss arguments {}
06:31:14 Added SoftmaxLogitsLoss with lambda = 1.0 and loss arguments {}
06:31:14 Added SoftmaxLogitsLoss with lambda = 1.0 and loss arguments {}
06:31:14 Built loss function
06:31:14 Built loss optimizer
06:31:14 Built loss scheduler
06:31:14 Built loss scheduler
06:31:14 Built loss scheduler
06:31:14 Loaded BaseStorage from ednaml.storage to build Storage
06:31:14 Loaded MultiClassificationTrainer from ednaml.trainer to build Trainer
06:31:14 Saving model metadata
06:31:14 Backing up metadata
06:31:14 Finished metadata backup
06:31:14 1 GPUs available
```

```
eml.train()
```

```
06:33:40 M F-Score      color: 0.702    vtype: 0.374    make: 0.242
06:33:40 W F-Score      color: 0.647    vtype: 0.323    make: 0.248
06:33:40 Saving model at save-frequency, at epoch 7, step 0
06:33:40 Saving model, optimizer, and scheduler.
06:33:52 ********** Completed epoch 8 **********
06:33:52 Model evaluation triggered, but gradients still need accumulation. Will evaluate after accumulation.
06:33:52 Model save triggered, but gradients still need accumulation. Will save after accumulation.
06:33:52 Parameter Group `opt-1`: Starting epoch 9 with 50 steps and learning rate 1.00000E-05
06:33:52 Evaluating model at test-frequency
06:33:55 Obtained features, validation in progress
06:33:55 Metrics        colorloss       typeloss        makeloss
06:33:55 Accuracy       color: 0.707    vtype: 0.384    make: 0.237
06:33:55 M F-Score      color: 0.707    vtype: 0.384    make: 0.237
06:33:55 W F-Score      color: 0.654    vtype: 0.352    make: 0.244
06:33:55 Saving model at save-frequency, at epoch 8, step 0
06:33:55 Saving model, optimizer, and scheduler.
06:34:07 ********** Completed epoch 9 **********
06:34:07 Model evaluation triggered, but gradients still need accumulation. Will evaluate after accumulation.
06:34:07 Model save triggered, but gradients still need accumulation. Will save after accumulation.
06:34:08 Parameter Group `opt-1`: Starting epoch 10 with 50 steps and learning rate 1.00000E-05
06:34:09 Evaluating model at test-frequency
06:34:12 Obtained features, validation in progress
06:34:12 Metrics        colorloss       typeloss        makeloss
06:34:12 Accuracy       color: 0.717    vtype: 0.374    make: 0.232
06:34:12 M F-Score      color: 0.717    vtype: 0.374    make: 0.232
06:34:12 W F-Score      color: 0.666    vtype: 0.342    make: 0.244
06:34:12 Saving model at save-frequency, at epoch 9, step 0
06:34:12 Saving model, optimizer, and scheduler.
06:34:24 ********** Completed epoch 10 **********
06:34:24 Model evaluation triggered, but gradients still need accumulation. Will evaluate after accumulation.
06:34:24 Model save triggered, but gradients still need accumulation. Will save after accumulation.
06:34:24 Final: Evaluating model at test-frequency
06:34:25 Obtained features, validation in progress
06:34:25 Metrics        colorloss       typeloss        makeloss
06:34:25 Accuracy       color: 0.722    vtype: 0.369    make: 0.217
06:34:25 M F-Score      color: 0.722    vtype: 0.369    make: 0.217
06:34:25 W F-Score      color: 0.675    vtype: 0.328    make: 0.223
06:34:25 Final: Saving model at save-frequency
06:34:25 Saving model, optimizer, and scheduler.
```

```
resp = eml.eval()
```

```
06:35:15 Obtained features, validation in progress
06:35:15 Metrics        colorloss       typeloss        makeloss
06:35:15 Accuracy       color: 0.722    vtype: 0.369    make: 0.217
06:35:15 M F-Score      color: 0.722    vtype: 0.369    make: 0.217
06:35:15 W F-Score      color: 0.675    vtype: 0.328    make: 0.223
```

5. Multibranch classification

Now we will try a model that uses multiple branches, each branch for a specific label, for classification. Then we will fuse the branches to classify one more things. So total, three classifications from a single model.

5.1 Vehicle color and type, fused to classify vehicle make

Now we will try a model that performs vehicle type AND vehicle color classification together, using 2 different branches, and fuses the results together for make classification. The config is already prepared for this in profiles/multibranch-ctm.yml

```
path_idx = 0          # Change this to whichever index in tuple has path
crawler_args = {"file_name" : "original_tool_image.zip"}
```

```
%load_ext autoreload
%autoreload 2
```

```
The autoreload extension is already loaded. To reload it, use:
  %reload_ext autoreload
```

```
from ednaml.core import EdnaML
from ednaml.generators import ClassificationGenerator

eml = EdnaML(config = ["./GLAMOR/profiles/CarZam/base_config.yml","./GLAMOR/profiles/CarZam/multibranch-ctm.yml"])

eml.cfg.EXECUTION.DATAREADER.CRAWLER_ARGS = crawler_args
eml.cfg.EXECUTION.DATAREADER.DATASET_ARGS["pathidx"] = path_idx
# We have already set these in config
#eml.cfg.EXECUTION.DATAREADER.DATASET_ARGS["annotationidx"] = class_idx
#eml.cfg.EXECUTION.DATAREADER.DATASET_ARGS["classificationclass"] = class_name
```

```
#eml.addGeneratorClass(ClassificationGenerator)
```

```
eml.apply()
```

```
- LOSS_OPTIMIZER
- LOSS_SCHEDULER
- LOGGING
- DEPLOYMENT
- MODEL_PLUGIN

06:36:52
06:36:52
06:36:52 ****************************************
06:36:52 No previous stop detected. Will start from epoch 0
06:36:52 Loaded BaseStorage from ednaml.storage to build Storage
06:36:52 Reading data with DataReader DataReader
06:36:52 Default CRAWLER is <class 'ednaml.crawlers.Crawler'>
06:36:52 Default DATASET is <class 'torch.utils.data.dataset.Dataset'>
06:36:52 Default GENERATOR is <class 'ednaml.generators.ImageGenerator.ImageGenerator'>
06:36:52 Updating GENERATOR using config specification to MultiClassificationGenerator
06:36:52 Updating CRAWLER to CarZamCrawler
06:36:52 Generated training data generator with 1575 training data points
06:36:52 Running classification model with classes: {'color': {'classes': 13}, 'vtype': {'classes': 13}, 'make': {'classes': 36}
06:36:52 Generated test data/query generator
06:36:52 Loaded multibranch_model_builder from ednaml.models to build model
Outliers

types: ['Convertible', 'PickupTruck', 'Hybrid', 'Van', 'ElectricVehicle', 'Diesel', 'Crossover', 'Wagon', 'LuxuryVehicle', 'Seda
Tuple_expanded

[('unzipped/original_tool_images/Hybrid Silver 2021 Toyota Avalon Hybrid.jpg', 2, 0, 2, 22), ('unzipped/original_tool_images/Cro
06:36:53 Finished instantiating model with MultiBranchResnet architecture
06:36:53 Adding plugins after constructing model
06:36:53 No saved model weights provided.
06:36:53 Model Summary retured the following error:
06:36:53 Traceback (most recent call last):
  File "/content/GLAMOR/src/ednaml/core/EdnaML.py", line 888, in getModelSummary
    self.cfg.TRAIN_TRANSFORMATION.INPUT_SIZE,
AttributeError: 'TransformationConfig' object has no attribute 'INPUT_SIZE'

06:36:53 Loaded ClassificationOptimizer from ednaml.optimizer to build Optimizer model
06:36:53 Built optimizer
06:36:53 Built scheduler
06:36:53 Added SoftmaxLogitsLoss with lambda = 1.0 and loss arguments {}
06:36:53 Added SoftmaxLogitsLoss with lambda = 1.0 and loss arguments {}
06:36:53 Added SoftmaxLogitsLoss with lambda = 1.0 and loss arguments {}
06:36:53 Added SoftmaxLogitsLoss with lambda = 1.0 and loss arguments {}
06:36:53 Added SoftmaxLogitsLoss with lambda = 1.0 and loss arguments {}
06:36:53 Built loss function
06:36:53 Built loss optimizer
06:36:53 Built loss scheduler
06:36:53 Built loss scheduler
06:36:53 Built loss scheduler
06:36:53 Built loss scheduler
06:36:53 Built loss scheduler
06:36:53 Loaded BaseStorage from ednaml.storage to build Storage
06:36:53 Loaded MultiBranchTrainer from ednaml.trainer to build Trainer
06:36:53 Saving model metadata
06:36:53 Backing up metadata
06:36:53 Finished metadata backup
06:36:53 1 GPUs available
```

```
eml.train()
```

```
06:37:05 Starting training
06:37:05 Logging to:    origtoolimgs-v1-multibranch-color-vtype-make-logger.log
06:37:05 Models will be saved to local directory:       origtoolimgs-v1-multibranch-color-vtype-make
06:37:05 Models will be saved with base name:   origtoolimgs-v1_epoch[].pth
06:37:05 Optimizers will be saved with base name:       origtoolimgs-v1_epoch[]_optimizer.pth
06:37:05 Schedulers will be saved with base name:       origtoolimgs-v1_epoch[]_scheduler.pth
06:37:05 Performing initial evaluation...
06:37:07 Obtained features, validation in progress
06:37:07 Metrics        color-fc        type-fc fuse    colorbranch     typebranch
06:37:07 Accuracy       color-fc: 0.086 type-fc: 0.071  fuse: 0.020     colorbranch: 0.005      typebranch: 0.040
06:37:07 M F-Score      color-fc: 0.086 type-fc: 0.071  fuse: 0.020     colorbranch: 0.005      typebranch: 0.040
06:37:07 W F-Score      color-fc: 0.045 type-fc: 0.042  fuse: 0.005     colorbranch: 0.006      typebranch: 0.013
06:37:07 Starting training from 0
06:37:07 Parameter Group `opt-1`: Starting epoch 0 with 50 steps and learning rate 1.00000E-05
06:37:21 ********** Completed epoch 0 **********
06:37:21 Model evaluation triggered, but gradients still need accumulation. Will evaluate after accumulation.
06:37:21 Model save triggered, but gradients still need accumulation. Will save after accumulation.
06:37:22 Parameter Group `opt-1`: Starting epoch 1 with 50 steps and learning rate 1.00000E-05
06:37:22 Evaluating model at test-frequency
06:37:25 Obtained features, validation in progress
06:37:25 Metrics        color-fc        type-fc fuse    colorbranch     typebranch
06:37:25 Accuracy       color-fc: 0.384 type-fc: 0.136  fuse: 0.106     colorbranch: 0.020      typebranch: 0.015
06:37:25 M F-Score      color-fc: 0.384 type-fc: 0.136  fuse: 0.106     colorbranch: 0.020      typebranch: 0.015
06:37:25 W F-Score      color-fc: 0.428 type-fc: 0.145  fuse: 0.104     colorbranch: 0.026      typebranch: 0.016
```

```
06:37:25 Saving model at save-frequency, at epoch 0, step 0
06:37:25 Saving model, optimizer, and scheduler.
06:37:38 ********** Completed epoch 1 **********
06:37:38 Model evaluation triggered, but gradients still need accumulation. Will evaluate after accumulation.
06:37:38 Model save triggered, but gradients still need accumulation. Will save after accumulation.
06:37:39 Parameter Group `opt-1`: Starting epoch 2 with 50 steps and learning rate 1.00000E-05
06:37:40 Evaluating model at test-frequency
06:37:42 Obtained features, validation in progress
06:37:42 Metrics        color-fc        type-fc fuse    colorbranch     typebranch
06:37:42 Accuracy       color-fc: 0.616 type-fc: 0.273  fuse: 0.177     colorbranch: 0.035      typebranch: 0.051
06:37:42 M F-Score      color-fc: 0.616 type-fc: 0.273  fuse: 0.177     colorbranch: 0.035      typebranch: 0.051
06:37:42 W F-Score      color-fc: 0.591 type-fc: 0.264  fuse: 0.177     colorbranch: 0.040      typebranch: 0.065
06:37:42 Saving model at save-frequency, at epoch 1, step 0
06:37:42 Saving model, optimizer, and scheduler.
06:37:55 ********** Completed epoch 2 **********
06:37:55 Model evaluation triggered, but gradients still need accumulation. Will evaluate after accumulation.
06:37:55 Model save triggered, but gradients still need accumulation. Will save after accumulation.
06:37:55 Parameter Group `opt-1`: Starting epoch 3 with 50 steps and learning rate 1.00000E-05
06:37:56 Evaluating model at test-frequency
06:37:58 Obtained features, validation in progress
06:37:58 Metrics        color-fc        type-fc fuse    colorbranch     typebranch
06:37:58 Accuracy       color-fc: 0.662 type-fc: 0.318  fuse: 0.232     colorbranch: 0.045      typebranch: 0.081
06:37:58 M F-Score      color-fc: 0.662 type-fc: 0.318  fuse: 0.232     colorbranch: 0.045      typebranch: 0.081
06:37:58 W F-Score      color-fc: 0.629 type-fc: 0.283  fuse: 0.227     colorbranch: 0.044      typebranch: 0.074
06:37:58 Saving model at save-frequency, at epoch 2, step 0
06:37:58 Saving model, optimizer, and scheduler.
06:38:12 ********** Completed epoch 3 **********
06:38:12 Model evaluation triggered, but gradients still need accumulation. Will evaluate after accumulation.
06:38:12 Model save triggered, but gradients still need accumulation. Will save after accumulation.
06:38:12 Parameter Group `opt-1`: Starting epoch 4 with 50 steps and learning rate 1.00000E-05
06:38:13 Evaluating model at test-frequency
06:38:16 Obtained features, validation in progress
06:38:16 Metrics        color-fc        type-fc fuse    colorbranch     typebranch
06:38:16 Accuracy       color-fc: 0.687 type-fc: 0.354  fuse: 0.273     colorbranch: 0.091      typebranch: 0.131
```

```
resp = eml.eval()
```

```
06:40:56 Obtained features, validation in progress
06:40:56 Metrics        color-fc        type-fc fuse    colorbranch     typebranch
06:40:56 Accuracy       color-fc: 0.707 type-fc: 0.379  fuse: 0.232     colorbranch: 0.106      typebranch: 0.152
06:40:56 M F-Score      color-fc: 0.707 type-fc: 0.379  fuse: 0.232     colorbranch: 0.106      typebranch: 0.152
06:40:56 W F-Score      color-fc: 0.650 type-fc: 0.295  fuse: 0.192     colorbranch: 0.065      typebranch: 0.120
```

✓ 0s completed at 7:54 AM