Weekly Progress Report

Date: [18/06/2023]

Introduction:

This Weekly Progress Report highlights the advancements made in computer vision tasks using the FastAI library. The report provides a comprehensive overview of the progress made during the week, including installation and setup, data loading and exploration, data preparation and processing, model creation and training, evaluation and metrics, predictions and analysis, and additional functionalities. The report concludes with reflections on the accomplishments and plans for future work.

1. Installation and Setup:

The week started with the installation of the FastBook library to ensure all the required dependencies were available. Following the successful installation, the necessary modules were imported, and the FastBook library was set up, ensuring a smooth workflow for the upcoming tasks.

2. Loading and Exploring Data:

To commence the data-related tasks, the Pascal VOC 2007 dataset was downloaded using the FastAI `untar_data` function. The dataset included various images and corresponding annotations. A CSV file, namely `train.csv`, was read from the dataset, and its contents were loaded into a DataFrame (`df`). This step allowed for an initial exploration of the data by displaying the first few rows of the DataFrame. The insights gained from this exploration provided valuable context for subsequent steps.

3. Data Preparation and Processing:

To prepare the data for model training, a `DataBlock` object was defined. The `DataBlock` facilitated the creation of a dataset by applying transformations and defining the data splitting strategy. By utilizing lambda functions and custom functions, the data transformations were designed to augment the dataset appropriately and ensure its compatibility with the subsequent model architecture.

4. Model Creation and Training:

The model creation and training phase began with the configuration of a `DataBlock` object. This `DataBlock` object was specifically tailored to handle the computer vision task at hand, incorporating image and multi-category blocks. The `DataBlock` object was then used to create dataloaders (`dls`), which efficiently loaded and preprocessed the data in preparation for model training.

A vision learner (`learn`) was instantiated, utilizing a pre-defined architecture (ResNet-18) and the previously created dataloaders. The model was then trained using the `fine_tune` method, specifying the number of epochs and learning rate. This step involved the iterative process of adjusting the model's weights to minimize the loss and optimize its performance on the given task.

5. Evaluation and Metrics:

To assess the trained model's performance, appropriate evaluation metrics were set up. These metrics included accuracy thresholds and other relevant measures depending on the specific task. The trained model was then validated using the validation dataset, allowing for an objective evaluation of its effectiveness. This step was essential for gauging the model's performance and identifying areas that required further refinement.

6. Predictions and Analysis:

To gain deeper insights into the model's performance and its ability to make predictions, the `get_preds` method was employed. This method provided access to the model's predicted outputs and the corresponding target labels. These outputs and labels were further analyzed using custom functions and techniques to evaluate the model's accuracy and gain a comprehensive understanding of its strengths and limitations.

Additional Functionalities:

Throughout the week, various additional functionalities were explored to enhance the workflow and expand the capabilities of the FastAI library. These functionalities included mathematical operations and the use of partial functions. By leveraging these advanced

techniques, the code showcased the library's versatility and flexibility in handling diverse computer vision tasks.

Code:

```
#hide ! [ -e /content ] && pip install -Uqq fastbook import fastbook fastbook.setup_book() #hide from fastbook import * from fastai.vision.all import * path = untar_data(URLs.PASCAL_2007) df = pd.read_csv(path/'train.csv') df.head() df.iloc[:,0] df.iloc[0,:] # Trailing :s are always optional (in numpy, pytorch, pandas, etc.), # so this is equivalent: df.iloc[0] df['fname'] tmp_df = pd.DataFrame({'a':[1,2], 'b':[3,4]}) tmp_df tmp_df['c'] = tmp_df['a']+tmp_df['b'] tmp_df dblock = DataBlock() dsets = dblock.datasets(df) len(dsets.train),len(dsets.valid) (4009, 1002) x,y = dsets.train[0] x,y x['fname'] dblock = DataBlock(get_x = lambda r: r['fname'], get_y = lambda r: r['labels']) dsets = dblock.datasets(df) dsets.train[0] def get_x(r): return r['fname'] def get_y(r): return r['labels'] dblock = DataBlock(get_x = get_x, get_y = get_y) dsets = dblock.datasets(df) dsets.train[0] def get_x(r): return path/'train'/r['fname'] def get_y(r): return r['labels'].split(' ') dblock = DataBlock(get_x = get_x, get_y = get_y) dsets = dblock.datasets(df) dsets.train[0] dblock = DataBlock(blocks=(ImageBlock, MultiCategoryBlock), get_x = get_x, get_y = get_y) dsets = dblock.datasets(df) dsets.train[0] idxs = torch.where(dsets.train[0][1]==1.)[0] dsets.train.vocab[idxs] def splitter(df): train = df.index[~df['is_valid']].tolist() valid = df.index[df['is_valid']].tolist() return train,valid dblock = DataBlock(blocks=(ImageBlock, MultiCategoryBlock), splitter=splitter, get_x=get_x, get_y=get_y) dsets = dblock.datasets(df) dsets.train[0] dblock = DataBlock(blocks=(ImageBlock, MultiCategoryBlock), splitter=splitter, get_x=get_x, get_y=get_y, item_tfms = RandomResizedCrop(128, min_scale=0.35)) dls = dblock.dataloaders(df) learn = vision_learner(dls, resnet18) x,y = to_cpu(dls.train.one_batch()) activs = learn.model(x) activs.shape activs[0] def binary_cross_entropy(inputs, targets): inputs = inputs.sigmoid() return -torch.where(targets==1, inputs, 1-inputs).log().mean() loss_func = nn.BCEWithLogitsLoss() loss = loss_func(activs, y) loss def say_hello(name, say_what="Hello"): return f"{say_what} {name}." say_hello('Jeremy'),say_hello('Jeremy', 'Ahoy!') f = partial(say_hello, say_what="Bonjour") f("Jeremy"),f("Sylvain") learn = vision_learner(dls, resnet50, metrics=partial(accuracy_multi, thresh=0.2)) learn.fine_tune(3, base_lr=3e-3, freeze_epochs=4) learn.metrics = partial(accuracy_multi, thresh=0.1) learn.validate() learn.metrics = partial(accuracy_multi, thresh=0.99) learn.validate() preds,targs = learn.get_preds() accuracy_multi(preds, targs, thresh=0.9, sigmoid=False) xs = torch.linspace(0.05,0.95,29) accs = [accuracy_multi(preds, targs, thresh=i, sigmoid=False) for i in xs] plt.plot(xs,accs); path = untar_data(URLs.BIWI_HEAD_POSE) #hide Path.BASE_PATH = path path.ls().sorted() (path/'01').ls().sorted() img_files = get_image_files(path) def img2pose(x): return Path(f'{str(x)[:-7]}pose.txt') img2pose(img_files[0]) im = PILImage.create(img_files[0]) im.shape im.to_thumb(160) cal = np.genfromtxt(path/'01'/'rgb.cal', skip_footer=6) def get_ctr(f): ctr = np.genfromtxt(img2pose(f), skip_header=3) c1 = ctr[0] * cal[0][0]/ctr[2] + cal[0][2] c2 = ctr[1] * cal[1][1]/ctr[2] + cal[1][2] return tensor([c1,c2]) get_ctr(img_files[0]) biwi =
```

```
DataBlock( blocks=(ImageBlock, PointBlock), get_items=get_image_files, get_y=get_ctr, splitter=FuncSplitter(lambda o: o.parent.name=='13'), batch_tfms=aug_transforms(size=(240,320)), dls = biwi.dataloaders(path) dls.show_batch(max_n=9, figsize=(8,6)) xb,yb = dls.one_batch() xb.shape,yb.shape yb[0] learn = vision_learner(dls, resnet18, y_range=(-1,1)) def sigmoid_range(x, lo, hi): return torch.sigmoid(x) * (hi-lo) + lo plot_function(partial(sigmoid_range,lo=-1,hi=1), min=-4, max=4) dls.loss_func learn.lr_find() lr = 1e-2 learn.fine_tune(3, lr) math.sqrt(0.0001) learn.show_results(ds_idx=1, nrows=3, figsize=(6,8))
```

Conclusion:

In conclusion, significant progress was made in computer vision tasks using the FastAI library during this week. The installation and setup were completed successfully, enabling seamless execution of subsequent tasks. The data loading and exploration phase provided valuable insights into the dataset, ensuring informed decision-making throughout the process. The data preparation and processing steps facilitated the transformation and splitting of the dataset,

preparing it for model training.

Model creation and training were executed efficiently, utilizing the power of the FastAI library. The trained model's performance was evaluated using appropriate metrics, and predictions were analyzed to gain insights into its accuracy and effectiveness. Additionally, the exploration of additional functionalities demonstrated the library's advanced capabilities.

Looking ahead, future work will involve further analysis of the results obtained, exploring additional functionalities and techniques provided by the FastAI library, and implementing improvements to enhance model performance. By building upon the foundations laid during this week, it is anticipated that more advanced computer vision tasks can be tackled and accomplished.

Overall, this week's progress has been substantial, and the obtained outcomes provide a solid foundation for future work. The experience gained through the practical implementation of the FastAI library will undoubtedly contribute to the successful completion of upcoming tasks and the achievement of project goals.

Thank you.

Best regards,

Shadab. A. Sheikh

References: https://github.com/fastai/fastbook/blob/master/06_multicat.ipynb