Weekly Progress Report

Date: [11/06/2023]

Project: Practical Deep Learning for Coders

This weekly progress report outlines the activities and achievements for the week. The focus was on working with Jupyter notebooks, fast.ai library, and deep learning concepts.

- **Fastai:** A learning courses and resources that offers practical deep, it provides a hands-on approach to learning deep learning, emphasizing practical implementation and real-world applications, working with Jupyter notebooks, fastai, and deep learning involves hands-on coding, understanding concepts related to neural networks, and addressing challenges specific to data analysis and model training.

- **Jupyter Notebook:** Got to know about visualizations, explanatory text, data analysis, scientific computing, and machine learning tasks. ex. Python, R, PyTorch.

- **Deep Learning:** machine learning that focuses on training artificial neural networks with multiple layers to learn representations of data. exposer on various domains, including computer vision, natural language processing, speech recognition.

- **Challenges:** When working with Jupyter notebooks, facing challenges debugging code errors, managing dependencies, handling large datasets efficiently, and ensuring reproducibility. It's important to carefully organize and document code in notebooks to maintain readability and facilitate collaboration. Additionally, deep learning itself can present challenges such as selecting appropriate architectures, dealing with overfitting or underfitting, and optimizing model performance.

Reference:- https://www.youtube.com/watch?v=0oyCUWLL_fU&t=15085s

- ## Introduction to Jupyter Notebooks:

Explored the functionalities and benefits of Jupyter notebooks.

Installed Jupyter notebook using the Google Colab platform.

Learned how to run code cells, add explanatory text, and visualize outputs.

Explored various markdown and code cell features to enhance documentation and code readability.

- **Setup and Installation:**

Installed necessary packages and libraries using pip, including fastbook, duckduckgo_search, jupyter_contrib_nbextensions, and nbdev.

Configured the fastbook library to facilitate the learning process.

Set up the Azure search key for image searching.

Image Searching and Data Acquisition:

Utilized the fast.ai and duckduckgo_search libraries to search for and download images.

Learned how to search for images using specific keywords, such as "grizzly bear."

Explored different image sources, such as Bing and DuckDuckGo, for retrieving relevant images.

Downloaded a set of images related to different types of bears (grizzly, black, and teddy) for further analysis.

- **Data Preprocessing and Visualization:**

Processed the downloaded images using various techniques, such as resizing and cropping.

Used the PIL (Python Imaging Library) to open and manipulate images.

Verified the integrity of the downloaded images and handled any failed images.

Created a DataBlock object to organize and preprocess the image data.

Visualized a batch of images using the DataLoaders object to ensure the correctness of the preprocessing steps.

- **Model Training and Evaluation:**

Constructed a deep learning model using the resnet18 architecture.

Split the data into training and validation sets using a random splitter.

Fine-tuned the model using transfer learning and the fast.ai framework.

Evaluated the model's performance using error rate and confusion matrix metrics.

Analyzed and interpreted the model's top losses to gain insights into the model's behavior.

- **Deployment and Interactivity:**

Exported the trained model for deployment using the learn.export() function.

Created an interactive interface to classify user-provided images.

Integrated widgets and buttons to enable image uploads and classification.

Utilized the trained model to predict the class and probability of the uploaded image.

Displayed the prediction and probability to the user using interactive widgets.

- **Challenges Faced:**

Debugging code errors and resolving package dependencies during installation.

Ensuring the availability and quality of the downloaded images for training.

Understanding and applying the fast.ai library and its associated functions effectively.

Managing and handling the interactive interface components for image classification.

- **Lessons Learned:**

Jupyter notebooks provide an interactive and flexible environment for deep learning projects.

The fast.ai library offers powerful tools and abstractions to simplify deep learning tasks.

Image preprocessing is crucial for model training and requires careful handling and verification.

Transfer learning and fine-tuning can greatly improve model performance and training efficiency.

Creating interactive interfaces enhances user experience and facilitates model deployment.

- **Next Steps:**

Explore additional deep learning architectures and techniques for further experimentation.

Extend the image classification project to handle a wider range of object categories.

Dive deeper into advanced topics such as natural language processing or generative models.

In conclusion, this week's progress was focused on getting started with Jupyter notebooks, utilizing the fast.ai library for deep learning tasks, and working through the process of image classification. The challenges faced were successfully addressed, and valuable lessons were learned throughout the week's activities. The next steps will involve further exploration and expansion of the project.

Please find attached a PDF file containing a summary of the progress made during this week, including code snippets, explanations, and visualizations.

Throughout the week, several challenges were encountered and successfully addressed. These challenges included debugging code errors and resolving package dependencies during the installation process. Additionally, ensuring the availability and quality of the downloaded images for training posed a challenge. Understanding and effectively utilizing the fast.ai library and its associated functions required focused efforts. Lastly, managing and handling the interactive interface components for image classification demanded careful attention, Engaging in the project activities.

Code:-

```
# -*- coding: utf-8 -*-
"""Untitled9.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1irB2Gkz3vaR1wnK7IScgS6Bdc7n8K8QW
"""

!pip install -Uqq fastbook duckduckgo_search
!pip install jupyter_contrib_nbextensions
!pip install nbdev

import fastbook
fastbook.setup_book()

from fastbook import *
from fastai.vision.widgets import *

key = os.environ.get('AZURE_SEARCH_KEY', 'XXX')

search_images_bing
```

```
search_images_ddg

doc(search_images_ddg)

from duckduckgo_search import ddg_images
def search_images(term, max_images=150):
  print(f"Searching for '{term}'")
  return L(ddg_images(term, max_results=max_images)).itemgot('image')

#results = search_images('grizzly bear')
results = search_images_ddg('grizzly bear', max_images=150)
len(results)

ims = ['http://3.bp.blogspot.com/-S1scRCkI3vY/UHzV2kucsPI/AAAAAAAAA-
k/YQ5UzHEm9Ss/s1600/Grizzly%2BBear%2BWildlife.jpg']

dest = 'images/grizzly.jpg'
download_url(ims[0], dest)

Path('images/grizzly.jpg')

im = Image.open(dest)
im.to_thumb(128,128)

bear_types = 'grizzly','black','teddy'
path = Path('bears')

doc(Path)

#!rm -rf /kaggle/working/bears
if not path.exists():
  path.mkdir()
  for o in bear_types:
    dest = (path/o)
    dest.mkdir(exist_ok=True)
    results = search_images_ddg(f'{o} bear', 150)
    print(results)
    download_images(dest, urls=results)

fns = get_image_files(path)
fns

??verify_images
failed = verify_images(fns)
failed

failed.map(Path.unlink);
```

```
??verify_images

class DataLoaders(GetAttr):
    def __init__(self, *loaders): self.loaders = loaders
    def __getitem__(self, i): return self.loaders[i]
    train,valid = add_props(lambda i,self: self[i])

bears = DataBlock(
    blocks=(ImageBlock, CategoryBlock),
    get_items=get_image_files,
    splitter=RandomSplitter(valid_pct=0.2, seed=42),
    get_y=parent_label,
    item_tfms=Resize(128))

doc(DataBlock)

blocks=(ImageBlock, CategoryBlock)

get_items=get_image_files

splitter=RandomSplitter(valid_pct=0.2, seed=42)

get_y=parent_label

item_tfms=Resize(128)

# Create a DataLoaders object from source
dls = bears.dataloaders(path)

dls.valid.show_batch(max_n=6, nrows=1)

doc(bears.new)
bears = bears.new(item_tfms=Resize(128, ResizeMethod.Squish))
dls = bears.dataloaders(path)
dls.valid.show_batch(max_n=6, nrows=2)

doc(bears.new)
bears = bears.new(item_tfms=Resize(128, ResizeMethod.Pad, pad_mode='zeros'))
dls = bears.dataloaders(path)
dls.valid.show_batch(max_n=6, nrows=2)

bears = bears.new(item_tfms=RandomResizedCrop(128, min_scale=0.3))
dls = bears.dataloaders(path)

doc(dls.train.show_batch)

# unique 會使用同張照片
dls.train.show_batch(max_n=3, nrows=1)
```

```
dls.train.show_batch(max_n=3, nrows=1, unique=True)

bears = bears.new(item_tfms=Resize(128), batch_tfms=aug_transforms(mult=2))
dls = bears.dataloaders(path)
dls.train.show_batch(max_n=8, nrows=2, unique=True)

bears = bears.new(
    item_tfms=RandomResizedCrop(128, min_scale=0.3),
    batch_tfms=aug_transforms())


dls = bears.dataloaders(path)

#dls.train.show_batch(max_n=8, nrows=2, unique=True)

learn = vision_learner(dls, resnet18, metrics=error_rate)
learn.fine_tune(4)

interp = ClassificationInterpretation.from_learner(learn)
interp.plot_confusion_matrix()

interp.plot_top_losses(3, nrows=2)

doc(interp.plot_top_losses)
??interp.plot_top_losses

# According to the matplot docs it's the image size in inches.
interp.plot_top_losses(3, nrows=2, figsize=(10,10))

#doc(ImageClassifierCleaner)
cleaner = ImageClassifierCleaner(learn)
cleaner

#hide
for idx in cleaner.delete(): cleaner.fns[idx].unlink()
for idx,cat in cleaner.change(): shutil.move(str(cleaner.fns[idx]), path/cat)

learn.export()

path = Path()
path.ls(file_exts='.pkl')

learn_inf = load_learner(path/'export.pkl')

learn_inf.predict('images/grizzly.jpg')

learn_inf.dls.vocab
```

```
#hide_output
btn_upload = widgets.FileUpload()
btn_upload

#hide
# For the book, we can't actually click an upload button, so we fake it
btn_upload = SimpleNamespace(data = ['images/grizzly.jpg'])

img = PILImage.create(btn_upload.data[-1])

#hide_output
out_pl = widgets.Output()
out_pl.clear_output()
with out_pl: display(img.to_thumb(128,128))
out_pl

pred,pred_idx,probs = learn_inf.predict(img.resize((128,128)))

#hide_output
lbl_pred = widgets.Label()
lbl_pred.value = f'Prediction: {pred}; Probability: {probs[pred_idx]:.04f}'
lbl_pred

#hide_output
btn_run = widgets.Button(description='Classify')
btn_run

def on_click_classify(change):
    img = PILImage.create(btn_upload.data[-1])
    out_pl.clear_output()
    with out_pl: display(img.to_thumb(128,128))
    pred,pred_idx,probs = learn_inf.predict(img)
    lbl_pred.value = f'Prediction: {pred}; Probability: {probs[pred_idx]:.04f}'

btn_run.on_click(on_click_classify)

#hide
#Putting back btn_upload to a widget for next cell
btn_upload = widgets.FileUpload()

#hide_output
VBox([widgets.Label('Select your bear!'),
      btn_upload, btn_run, out_pl, lbl_pred])
```

Thank you.

Sincerely, Shadab.A. Sheikh