Multi-Task Knowledge Distillation for Eye Disease Prediction

Our implementation of knowledge distillation on multi task learning

Malwina Wojewoda, Mateusz Sperkowski, Szymon Rećko

Warsaw University of Technology

May 24, 2022

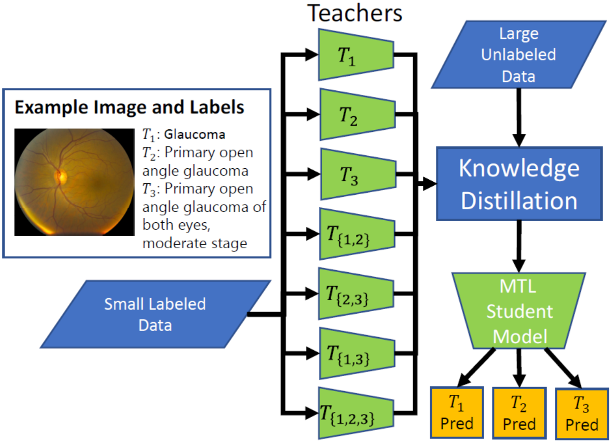
Wydział Matematyki i Nauk Informacyjnych
POLITECHNIKA WARSZAWSKA

Table of Contents

Paper

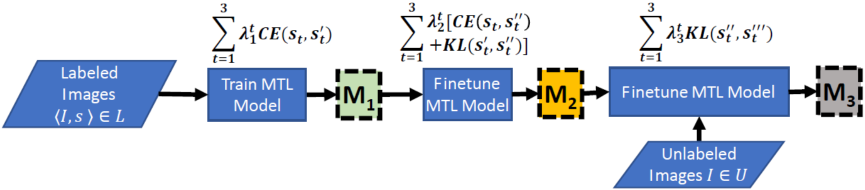
Given a fundus image, authors of [Chelaramani et al., 2021] aim to evaluate various solutions for learning deep neural classifiers using small labeled data for three tasks related to eye disease prediction.The problem is challenging because of small data size, need for predictions across multiple tasks, handling image variations, and large number of hyper-parameter choices. Their solution is to create MTL-based teacher ensemble method for knowledge distillation.

Reminder



Wydział Matematyki i Nauk Informacyjnych
POLITECHNIKA WARSZAWSKA

Architecture



Wydział Matematyki i Nauk Informacyjnych
POLITECHNIKA WARSZAWSKA

An overview of our progress

What we did so far:

- Data preprocessing - Mateusz
- Create Datasets and Models classes - Szymon
- Selecting Subset of tasks to work on - together
- Creating MTL teachers model and predicting on the same dataset - Malwina
- Transfer learning to student model - Malwina

Dataset

```python
class IDRiD_Dataset(Dataset):
    def __init__(self, transform, data_type="train"):
        #path to images
        path2data = os.path.join(path2img,data_type)
        #list of images
        filenames = os.listdir(path2data)
        #fullpath
        self.full_filenames = [os.path.join(path2data,f) for f in filenames]
        #labels
        csv_filename="labels.csv"
        path2csvLabels = os.path.join(path2labels,data_type,csv_filename)
        labels_df = pd.read_csv(path2csvLabels,index_col=[0])
        self.labels = labels_df.iloc[:,1:]
        self.transform = transform

    def __len__(self):
        return len(self.full_filenames)

    def __getitem__(self,idx):
        image = Image.open(self.full_filenames[idx])
        image = self.transform(image)
        table = self.labels.loc[idx].to_numpy()
        return image, table[0],table[1],table[2:4],table[4:6]

```

Model

```python
class MTL(nn.Module):
    def __init__(self):
        super(MTL, self).__init__()
        resnet50 = models.resnet50(pretrained=True)
        self.features = torch.nn.Sequential(*(list(resnet50.children())[:-1]))
        self.last = nn.Sequential(nn.Linear(2048, 1024),nn.ReLU())
        self.retinopathy_classifier = nn.Sequential(nn.Linear(1024, 512),nn.ReLU(), nn.Linear(512, 5))
        self.macular_edema_classifier = nn.Sequential(nn.Linear(1024, 512),nn.ReLU(), nn.Linear(512, 5))
        self.fovea_center_cords = nn.Sequential(nn.Linear(1024, 512),nn.ReLU(), nn.Linear(512, 2))
        self.optical_disk_cords = nn.Sequential(nn.Linear(1024, 512),nn.ReLU(), nn.Linear(512, 2))


    def forward(self, data):
        out = self.features.forward(data).squeeze()
        out = self.last.forward(out)
        return (self.retinopathy_classifier(out),
                self.macular_edema_classifier(out),
                self.fovea_center_cords(out),
                self.optical_disk_cords(out))
```

Wydział Matematyki i Nauk Informacyjnych
POLITECHNIKA WARSZAWSKA

Training loop

```
def fit_iter(self, train_dl):
    train_loss = 0.0
    loss0sum = 0.0
    loss1sum = 0.0
    loss2sum = 0.0
    loss3sum = 0.0
    loss = torch.tensor(0)
    accuracy0 = 0.0
    accuracy1 = 0.0
    for i, (imgs, retinopathy_label, macular_edema_label, fovea_center_labels, optical_disk_labels) in enumerate(train_dl):
        fovea_center_labels[:0], fovea_center_labels[:1] = fovea_center_labels[:0]*Rx, fovea_center_labels[:1]*Ry
        optical_disk_labels[:0], optical_disk_labels[:1] = optical_disk_labels[:0]*Rx, optical_disk_labels[:1]*Ry
        self.optimizer.zero_grad()
        retinopathy_pred, macular_edema_pred, fovea_center_pred, optical_disk_pred = self.forward(imgs.to(device))
        loss0 = self.criterion(retinopathy_pred, retinopathy_label.to(device).to(torch.int64)).to(torch.float64)*10
        loss1 = self.criterion(macular_edema_pred, macular_edema_label.to(device).to(torch.int64)).to(torch.float64)*10
        loss2 = torch.sqrt(self.criterion2(fovea_center_pred.to(torch.double),fovea_center_labels.to(device).to(torch.double)))/10
        loss3 = torch.sqrt(self.criterion2(optical_disk_pred.to(torch.double),optical_disk_labels.to(device).to(torch.double)))/10
        loss0sum += loss0
        loss1sum += loss1
        loss2sum += loss2
        loss3sum += loss3
        pred0 = F.softmax(retinopathy_pred, dim = -1).argmax(dim=-1)
        accuracy0 += pred0.eq(retinopathy_label.to(device)).sum().item()
        pred1 = F.softmax(macular_edema_pred, dim = -1).argmax(dim=-1)
        accuracy1 += pred1.eq(macular_edema_label.to(device)).sum().item()
        loss = torch.stack((loss0, loss1, loss2 ,loss3))[self.tasks].sum()
        loss.backward()
        self.optimizer.step()
        train_loss += loss
    print("\nTotal Loss: {}\nLoss0: {}  Accuracy0: {}\nLoss1: {}  Accuracy1: {}\nLoss2: {}\nLoss3: {}".format(train_loss, loss0sum,
    return train_loss
```

Wydział Matematyki i Nauk Informacyjnych
POLITECHNIKA WARSZAWSKA

Settings

```
rx=450
ry=300
old_x=4288
old_y=2848
Rx=rx/old_x
Ry=ry/old_y

data_transformer = transforms.Compose([transforms.Resize((rx,ry)),transforms.ToTensor(),
                                        transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])])
train_ds = IDRiD_Dataset(data_transformer,'train')
train_dl = DataLoader(train_ds,batch_size=32,shuffle=True)
mtl = MTL()

if torch.cuda.is_available():
    device = torch.device("cuda")
    mtl=mtl.to(device)

criterion = nn.CrossEntropyLoss()
optimizer = torch.optim.SGD(mtl.parameters(),
                                weight_decay=1e-6,
                                momentum=0.9,
                                lr=1e-3,
                                nesterov=True)
scheduler = ReduceLROnPlateau(optimizer,
                                factor=0.5,
                                patience=3,
                                min_lr=1e-7,
                                verbose=True)
tasks=[[0, 1, 2]]
```

Wydział Matematyki i Nauk Informacyjnych
POLITECHNIKA WARSZAWSKA

Division of future works

Our plans for the future:

- Resize and save dataset - Szymon
- Predict on the Unlabeled dataset and teach last model. - Mateusz
- Add dual cost at M2. - Szymon
- Create some ensemble of teachers and repeat the process. - Malwina

Wydział Matematyki i Nauk Informacyjnych
POLITECHNIKA WARSZAWSKA

Possible future expansions of project

Depending on the timeline of the project and ongoing problems we might expand to following fields:

- Expand teacher models
- Explore possible teacher ensembles
- Add freezing to the first layers

References I

Chelaramani, S., Gupta, M., Agarwal, V., Gupta, P., and Habash, R. (2021).
Multi-task knowledge distillation for eye disease prediction.
In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 3983–3993.