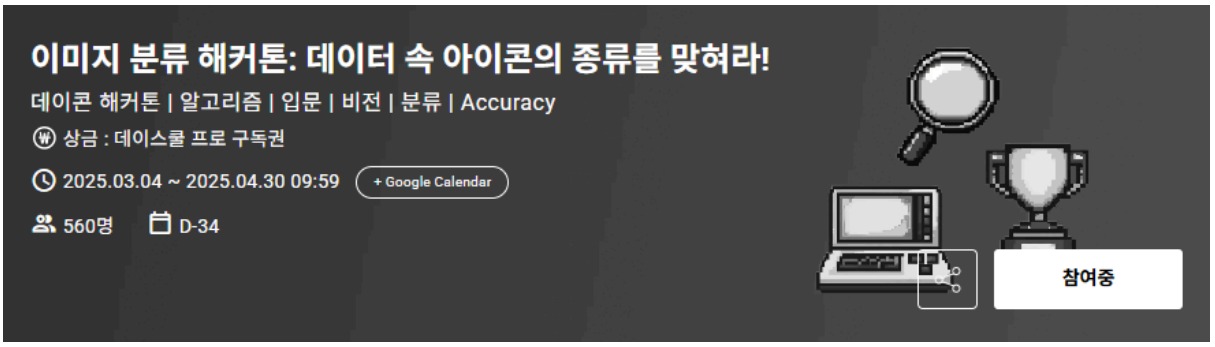
03.27 스터디

주제: 데이콘 이미지 분류 헤커톤



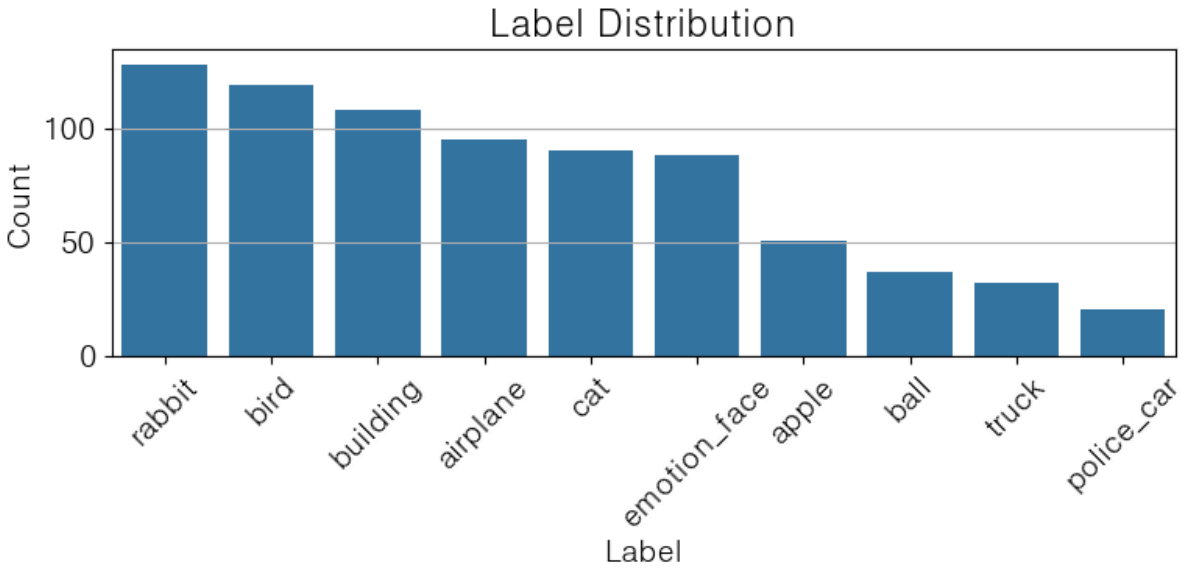
모델 input

- 이미지 크기 32×23
- 이미지 채널 Grayscale (1채널)
- 데이터 형식 .csv 파일 (train.csv, test.csv)
  - train.csv ID,label,0,1,2,...,1023
  - test.csv ID,0,1,2,...,1023 (라벨 없음)

모델 Output

- 10가지 아이콘의 종류(airplane, apple, ball, bird, building, cat, emotion_face, police_car, rabbit, truck) 중 하나로 분류

train.csv 라벨 분포



결과(1차)

모델명	적용기법	acc
model_1	단순 CNN	0.80519
model_2	스케줄러	0.81169
model_3	드롭아웃 + 배치정규화	0.93506
model_4	데이터 증강	0.90260

코드

```
class model_3(nn.Module):
    def __init__(self, num_classes):
        super().__init__()
        self.relu = nn.ReLU(inplace=True)
        self.maxpool = nn.MaxPool2d(2, 2)
        self.flatten = nn.Flatten()

        self.features = nn.Sequential(
            nn.Conv2d(1, 32, 3, padding=1),
            nn.BatchNorm2d(32),
            self.relu,
```

```
            nn.Conv2d(32, 32, 3, padding=1),
            nn.BatchNorm2d(32),
            self.relu,
            self.maxpool,
            nn.Dropout(0.2),

            nn.Conv2d(32, 64, 3, padding=1),
            nn.BatchNorm2d(64),
            self.relu,
            nn.Conv2d(64, 64, 3, padding=1),
            nn.BatchNorm2d(64),
            self.relu,
            self.maxpool,
            nn.Dropout(0.3),

            nn.Conv2d(64, 128, 3, padding=1),
            nn.BatchNorm2d(128),
            self.relu,
            nn.Conv2d(128, 128, 3, padding=1),
            nn.BatchNorm2d(128),
            self.relu,
            self.maxpool,
            nn.Dropout(0.4)
        )

        self.classifier = nn.Sequential(
            nn.Linear(128 * 4 * 4, 256),
            self.relu,
            nn.Dropout(0.4),
            nn.Linear(256, num_classes)
        )

    def forward(self, x):
        x = self.features(x)
        x = self.flatten(x)
        x = self.classifier(x)
        return x
```

```
class ImageCSVLoader(Dataset):
    def __init__(self, df, is_test=False):
        self.df = df.reset_index(drop=True)
        self.is_test = is_test

        self.images = self.df.iloc[:, 1:].values if is_test else self.df.iloc[:, 2:].values
        self.images = self.images.reshape(-1, 1, 32, 32).astype(np.float32) / 255.(

        if not is_test:
            self.labels = self.df['label'].values.astype(np.int64)  # 이미 정수임

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

    def __getitem__(self, idx):
        img = torch.tensor(self.images[idx])
        if self.is_test:
            return img
        label = torch.tensor(self.labels[idx])
        return img, label
```

```
full_train_df = pd.read_csv('./data/train.csv')

label_encoder = LabelEncoder()
full_train_df['label'] = label_encoder.fit_transform(full_train_df['label'])

dataset = ImageCSVLoader(full_train_df, is_test=False)

train_size = int(0.8 * len(dataset))
val_size = len(dataset) - train_size
train_ds, val_ds = random_split(dataset, [train_size, val_size])

batch_size = 64
train_loader = DataLoader(train_ds, batch_size=batch_size, shuffle=True)
val_loader = DataLoader(val_ds, batch_size=batch_size, shuffle=False)

device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
num_classes = len(label_encoder.classes_)
```

```
model = model_3(num_classes=num_classes).to(device)
criterion = torch.nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(model.parameters(), lr=0.001)
scheduler = StepLR(optimizer, step_size=5, gamma=0.9)
history = np.zeros((0, 5))
num_epochs = 30

history = fit(model, optimizer, criterion, num_epochs, train_loader, val_loader,
```

스코어

개선점

클래스별 정확도 히트맵 만들어보기

학습 데이터수가 적은 클래스 위주로 데이터 증강

모델 아키텍처 가져오기