Analysis of Visual Mandela Effect

Team 8
2021320086 김주연
2023320150 강지민

Problem Definition

Previous Goal: Predicting Student Depression

Problems

- Topic overlapped with other team
- The dataset we chose did not accurately reflect the real world.

Problem Definition

Previous Goal: Predicting Student Depression

Problems

- Topic overlapped with other team
- The dataset we chose did not accurately reflect the real world.

⬇

New Goal: **Analysis of Visual Mandela Effect**

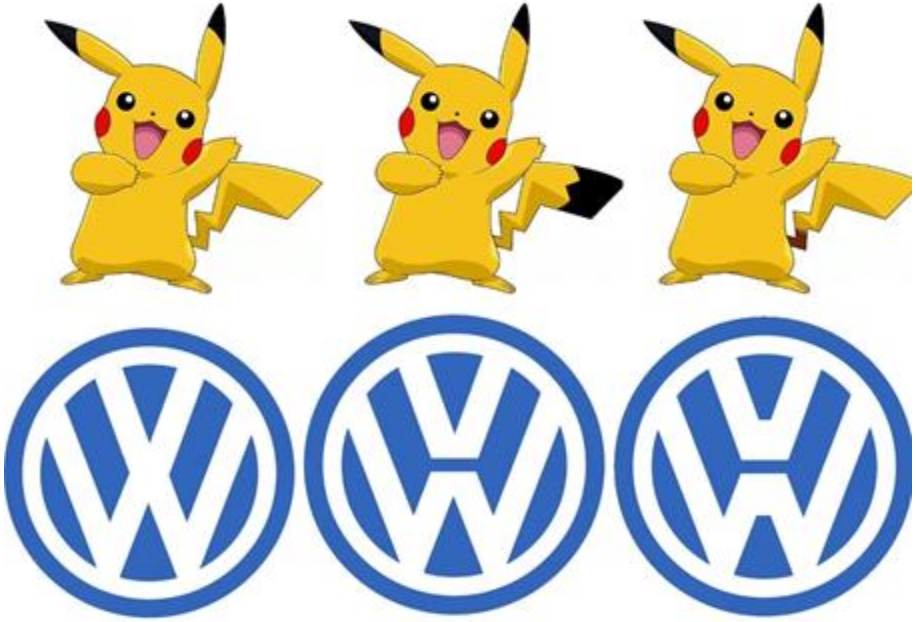
Mandela Effect

Shared False Memory across People

- Many people remember Mandela died at prison
- but in fact, he became president after he was imprisoned and lived till 2013.

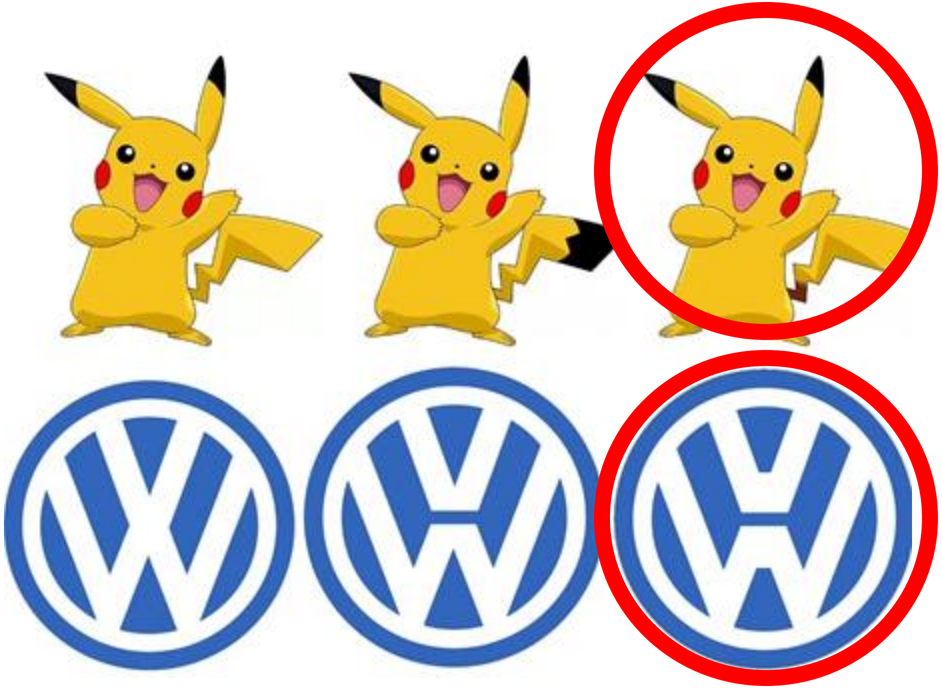
Our Topic - Visual Mandela Effect

What's The Original One?



Our Topic - Visual Mandela Effect

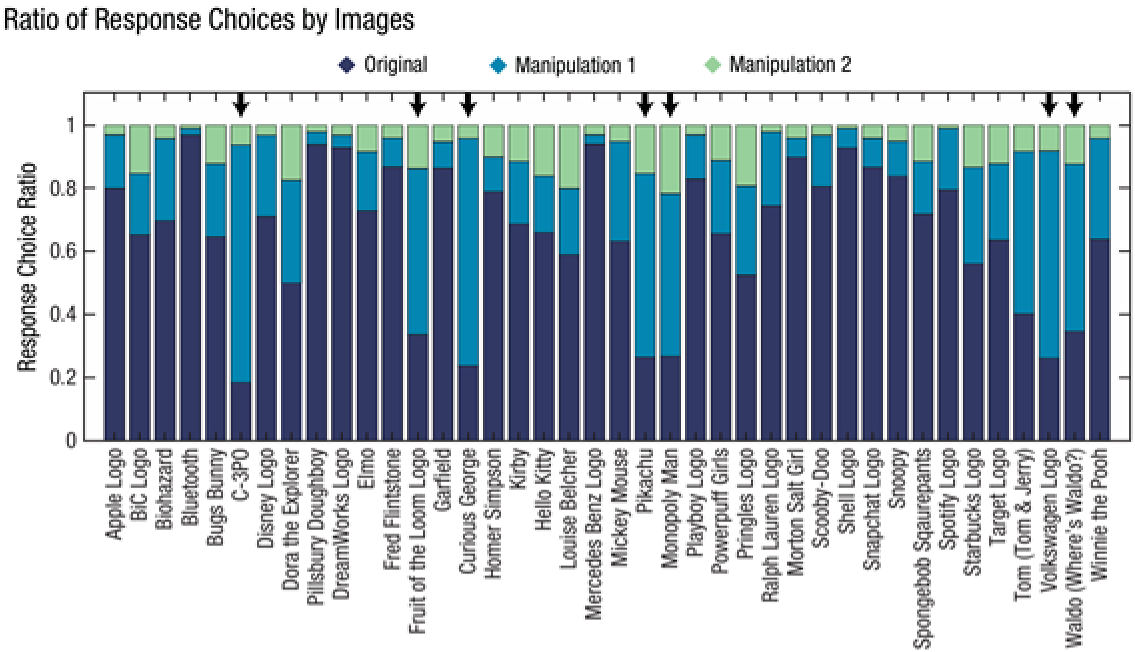
What's The Original One?



Our Goal - Analysing VME Using DS/ML

Significance Comparison of VME / non-VME

a



- Are there shared features across VME images?
- If so, Can we analyze using DS/ML?

Prasad, D., & Bainbridge, W. A. (2022).

Technical Definition & Result

0. Load Dataset & Preprocessing

The Visual Mandela Effect: Evidence for ... | Metadata | Files | Components | Analytics | Registrations

Notice: There was a service disruption event on Friday, May 23 that may have affected your OSF activity. Please see https://www.cos.io/about/news/notice-of-osf-service-disruption for more information

463.4MB | Public | 0 | ...

The Visual Mandela Effect: Evidence for Shared Specific False Memories

Dataset is public in https://osf.io/7cmwf/

Build dataframe

```python
# 1) Configuration
root_dir = Path('drive/MyDrive/Colab Notebooks/datas/Stimulus Images')
n_splits = 8
batch_size = 16
num_epochs = 5
learning_rate = 1e-4
device = "cuda" if torch.cuda.is_available() else "cpu"

# 2) Build DataFrame of image paths, categories, labels
records = []
for label_dir in root_dir.iterdir():
    if label_dir.is_dir():
        label = 1 if label_dir.name.lower() == "vme" else 0
        for img_path in label_dir.glob("*.jpg"):
            # 예: 'Apple_Original.jpg' → 'Apple'
            category = img_path.stem.split('_')[0]
            records.append({
                'path': str(img_path),
                'category': category,
                'label': label
            })
df = pd.DataFrame(records)
```

Preprocess

```python
# 3) Transforms
train_tf = transforms.Compose([
    transforms.RandomResizedCrop(224),
    transforms.RandomHorizontalFlip(),
    transforms.ColorJitter(0.2,0.2,0.2,0.1),
    transforms.ToTensor(),
    transforms.Normalize((0.4815,0.4578,0.4082),(0.2686,0.2613,0.2758))
])
val_tf = transforms.Compose([
    transforms.Resize(256),
    transforms.CenterCrop(224),
    transforms.ToTensor(),
    transforms.Normalize((0.4815,0.4578,0.4082),(0.2686,0.2613,0.2758))
])
```

Load data

```python
# 4) Dataset
class VMEDataset(Dataset):
    def __init__(self, df, transform):
        self.df = df.reset_index(drop=True)
        self.transform = transform
    def __len__(self): return len(self.df)
    def __getitem__(self, idx):
        row = self.df.iloc[idx]
        img = Image.open(row.path).convert("RGB")
        img = self.transform(img)
        label = torch.tensor(row.label, dtype=torch.float32)
        cat = row.category
        return img, label, cat

skf = StratifiedKFold(n_splits=8, shuffle=True, random_state=42)

for fold, (train_idx, val_idx) in enumerate(
        skf.split(df, df['label']), start=1):

    train_df = df.iloc[train_idx].reset_index(drop=True)
    val_df   = df.iloc[val_idx].reset_index(drop=True)

    train_ds = VMEDataset(train_df, train_tf)
    val_ds   = VMEDataset(val_df,   val_tf)

    train_loader = DataLoader(train_ds, batch_size=16,
                              shuffle=True,  num_workers=4)
    val_loader   = DataLoader(val_ds,   batch_size=16,
                              shuffle=False, num_workers=4)
```

Technical Definition & Result

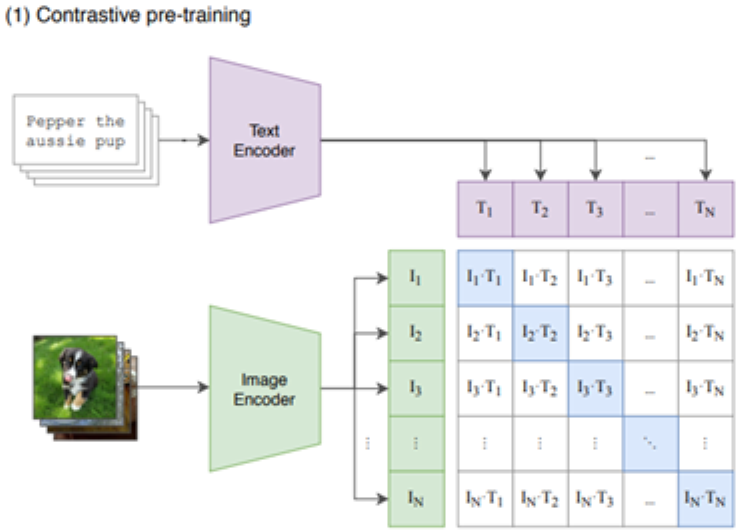
1. Contrastive Language-Image Pretrained

```python
# 6) Model definition
class CLIPFusionClassifier(nn.Module):
    def __init__(self, clip_model, hidden_dim=256):
        super().__init__()
        self.clip = clip_model
        embed_dim = clip_model.visual.output_dim
        self.img_proj = nn.Linear(embed_dim, embed_dim)
        self.txt_proj = nn.Linear(embed_dim, embed_dim)
        self.fuse     = nn.Sequential(
            nn.Linear(embed_dim*2, hidden_dim),
            nn.ReLU(),
            nn.Dropout(0.2),
            nn.Linear(hidden_dim, 1)
        )

    def forward(self, images, text_tokens):
        img_emb = self.img_proj(self.clip.encode_image(images).float())
        txt_emb = self.txt_proj(self.clip.encode_text(text_tokens).float())
        fused   = torch.cat([img_emb, txt_emb], dim=-1)
        logits  = self.fuse(fused).squeeze(-1)
        return fused, logits

# 7) Load CLIP and instantiate model
clip_model, preprocess = clip.load("RN50", device=device)
model = CLIPFusionClassifier(clip_model).to(device)
for p in model.clip.parameters(): p.requires_grad = False
optimizer = optim.Adam(filter(lambda p: p.requires_grad, model.parameters()), lr=learning_rate)
criterion = nn.BCEWithLogitsLoss()
```

Image + Language prompt
('a clear icon of {}')



Technical Definition & Result

2. 8-fold cross validation

Training step

```python
# 8) Cross-validation training
skf = StratifiedKFold(n_splits=n_splits, shuffle=True, random_state=42)
fold_metrics = []
for fold, (train_idx, val_idx) in enumerate(skf.split(df, df['label']), start=1):
    print(f"=== Fold {fold}/{n_splits} ===")
    train_df = df.iloc[train_idx].reset_index(drop=True)
    val_df   = df.iloc[val_idx].reset_index(drop=True)

    train_ds = VMEDataset(train_df, train_tf)
    val_ds   = VMEDataset(val_df, val_tf)
    train_loader = DataLoader(train_ds, batch_size=batch_size, shuffle=True)
    val_loader   = DataLoader(val_ds, batch_size=batch_size, shuffle=False)

    # Train epochs
    for epoch in range(num_epochs):
        model.train()
        running_loss = 0.0
        for imgs, labels, cats in train_loader:
            imgs = imgs.to(device)
            labels = labels.to(device)
            tokens = clip.tokenize([make_prompt(c) for c in cats]).to(device)

            _, logits = model(imgs, tokens)
            loss = criterion(logits, labels)
            running_loss += loss.item() * imgs.size(0)

            optimizer.zero_grad()
            loss.backward()
            optimizer.step()
        epoch_loss = running_loss / len(train_ds)
        print(f" Epoch {epoch+1}/{num_epochs} | Train Loss: {epoch_loss:.4f}")
```

Validation step

```python
# Validation
model.eval()
ys, ps, preds = [], [], []
with torch.no_grad():
    for imgs, labels, cats in val_loader:
        imgs=imgs.to(device)
        labels=labels.to(device)
        tokens=clip.tokenize([make_prompt(c) for c in cats]).to(device)

        _, logits=model(imgs, tokens)
        probs = torch.sigmoid(logits)

        ys.extend(labels.cpu().numpy())
        ps.extend(probs.cpu().numpy())
        preds.extend((probs.cpu().numpy() >= 0.5).astype(int))

auc = roc_auc_score(ys, ps)
preds = (np.array(ps)>=0.5).astype(int)
f1 = f1_score(ys, preds)
acc  = accuracy_score(ys, preds)
print(f"Fold {fold+1} AUC: {auc:.3f}, F1: {f1:.3f}, Accuracy: {acc:.3f}")
fold_metrics.append((auc, f1))
```

Technical Definition&Result

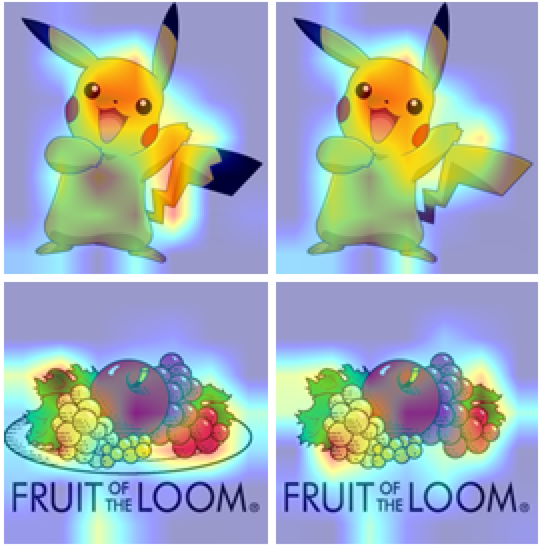
2. 8-fold cross validation - Result

```
=== Fold 1/8 ===
 Epoch 1/5 - Train Loss: 0.6152
 Epoch 2/5 - Train Loss: 0.5100
 Epoch 3/5 - Train Loss: 0.4711
 Epoch 4/5 - Train Loss: 0.4420
 Epoch 5/5 - Train Loss: 0.4030
Fold 2 AUC: 1.000, F1: 0.000, Accuracy: 0.800
=== Fold 2/8 ===
 Epoch 1/5 - Train Loss: 0.3744
 Epoch 2/5 - Train Loss: 0.3362
 Epoch 3/5 - Train Loss: 0.2860
 Epoch 4/5 - Train Loss: 0.2307
 Epoch 5/5 - Train Loss: 0.1831
Fold 3 AUC: 1.000, F1: 0.500, Accuracy: 0.867
=== Fold 3/8 ===
 Epoch 1/5 - Train Loss: 0.1537
 Epoch 2/5 - Train Loss: 0.1166
 Epoch 3/5 - Train Loss: 0.0885
 Epoch 4/5 - Train Loss: 0.0633
 Epoch 5/5 - Train Loss: 0.0482
Fold 4 AUC: 1.000, F1: 1.000, Accuracy: 1.000
=== Fold 4/8 ===
 Epoch 1/5 - Train Loss: 0.0361
 Epoch 2/5 - Train Loss: 0.0285
 Epoch 3/5 - Train Loss: 0.0224
 Epoch 4/5 - Train Loss: 0.0178
 Epoch 5/5 - Train Loss: 0.0153
```

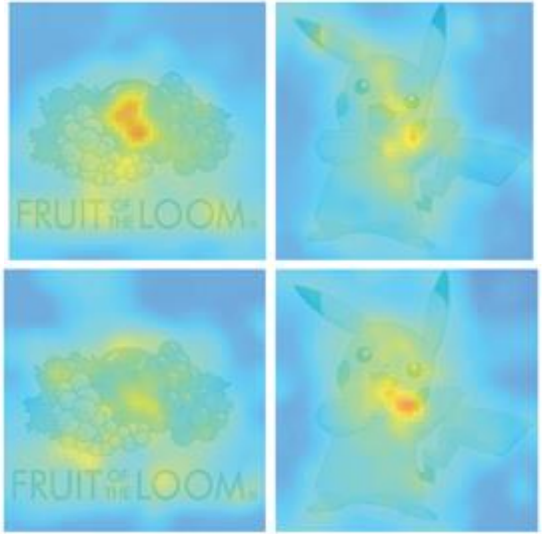
```
=== Fold 5/8 ===
 Epoch 1/5 - Train Loss: 0.0116
 Epoch 2/5 - Train Loss: 0.0102
 Epoch 3/5 - Train Loss: 0.0097
 Epoch 4/5 - Train Loss: 0.0073
 Epoch 5/5 - Train Loss: 0.0065
Fold 6 AUC: 1.000, F1: 1.000, Accuracy: 1.000
=== Fold 6/8 ===
 Epoch 1/5 - Train Loss: 0.0059
 Epoch 2/5 - Train Loss: 0.0058
 Epoch 3/5 - Train Loss: 0.0052
 Epoch 4/5 - Train Loss: 0.0043
 Epoch 5/5 - Train Loss: 0.0041
Fold 7 AUC: 1.000, F1: 1.000, Accuracy: 1.000
=== Fold 7/8 ===
 Epoch 1/5 - Train Loss: 0.0041
 Epoch 2/5 - Train Loss: 0.0032
 Epoch 3/5 - Train Loss: 0.0030
 Epoch 4/5 - Train Loss: 0.0029
 Epoch 5/5 - Train Loss: 0.0027
Fold 8 AUC: 1.000, F1: 1.000, Accuracy: 1.000
=== Fold 8/8 ===
 Epoch 1/5 - Train Loss: 0.0026
 Epoch 2/5 - Train Loss: 0.0024
 Epoch 3/5 - Train Loss: 0.0023
 Epoch 4/5 - Train Loss: 0.0021
 Epoch 5/5 - Train Loss: 0.0020
Fold 9 AUC: 1.000, F1: 1.000, Accuracy: 1.000
```

Technical Definition&Result

3. GradCam Heat map visualization & Comparison with behavioural heat map



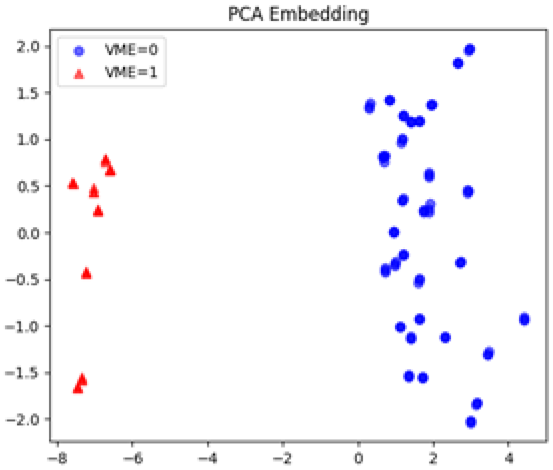
Our heatmap using GradCam



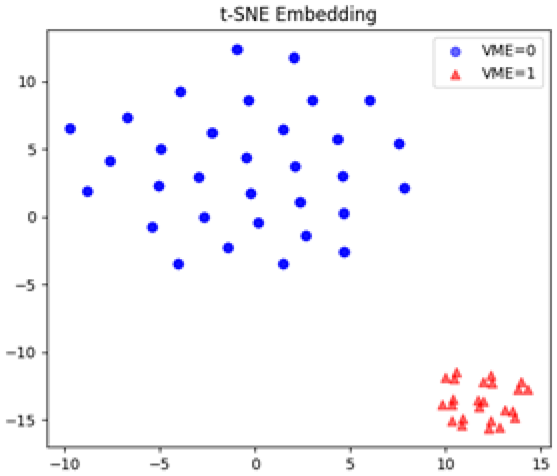
Behavioural heatmap
from Prasad, D., & Bainbridge, W. A. (2022).

Technical Definition&Result

4. Feature projection using PCA&t-SNE



PCA



t-SNE

Conclusion

Significance of This Experiment

1. Showed that a ML model can distinguish between VME-eliciting and non-VME eliciting images.
2. Showed another approach to studying VME.
3. Showed the possibility of automated discovery.

Conclusion

Limitations

1. VME is about microscopic, detail-level mismatches, but our model and processing pipeline do not explicitly preserve these features.
2. The dataset was too small, due to lack of study on VME.