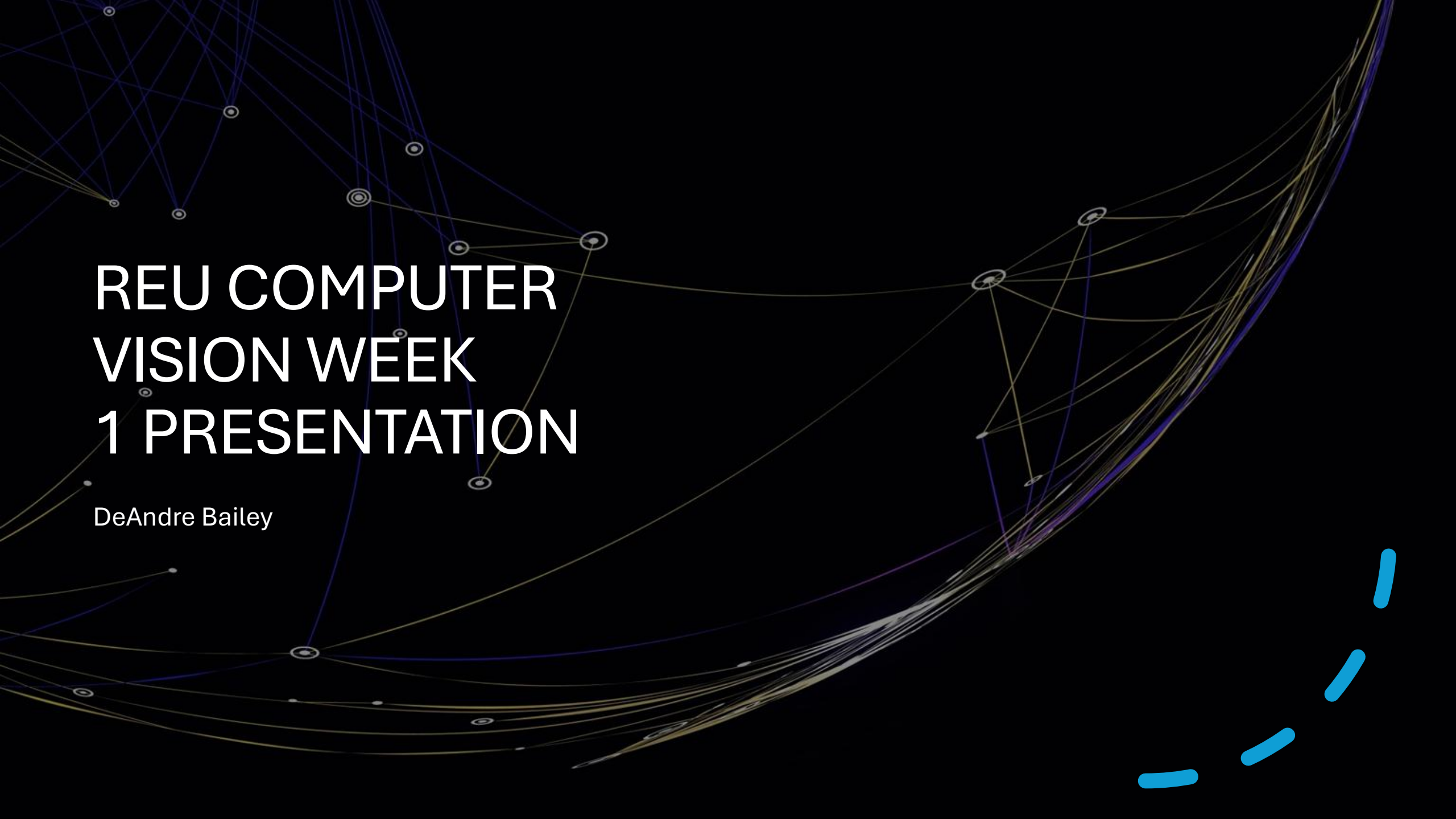


REU COMPUTER VISION WEEK 1 PRESENTATION

DeAndre Bailey

INTRODUCTION

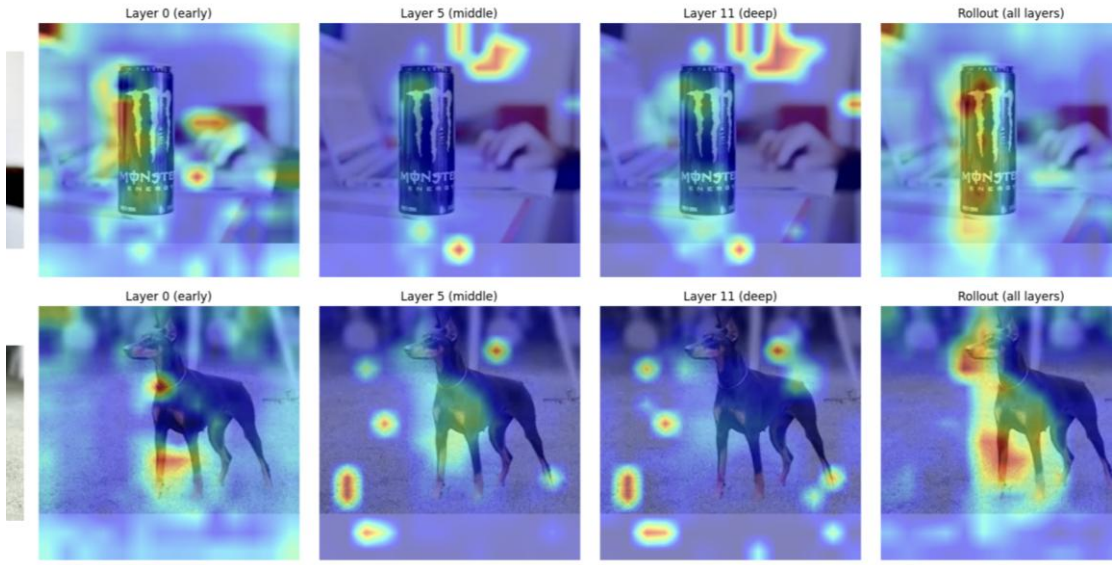
- DeAndre Bailey
- University of Central Florida, BS in Computer Science
- No Research Experience
- Enthusiastic about using AI to Help People

ASSIGNMENT 1

- Introduction to Coding ViT
- Calculating Predictions
- Attention Maps
- Rollout Grid
- Report

ASSIGNMENT 1 RESULTS AND FINDINGS

- Classification Issues
- Registers
- Layer 11 vs Rollout Grid



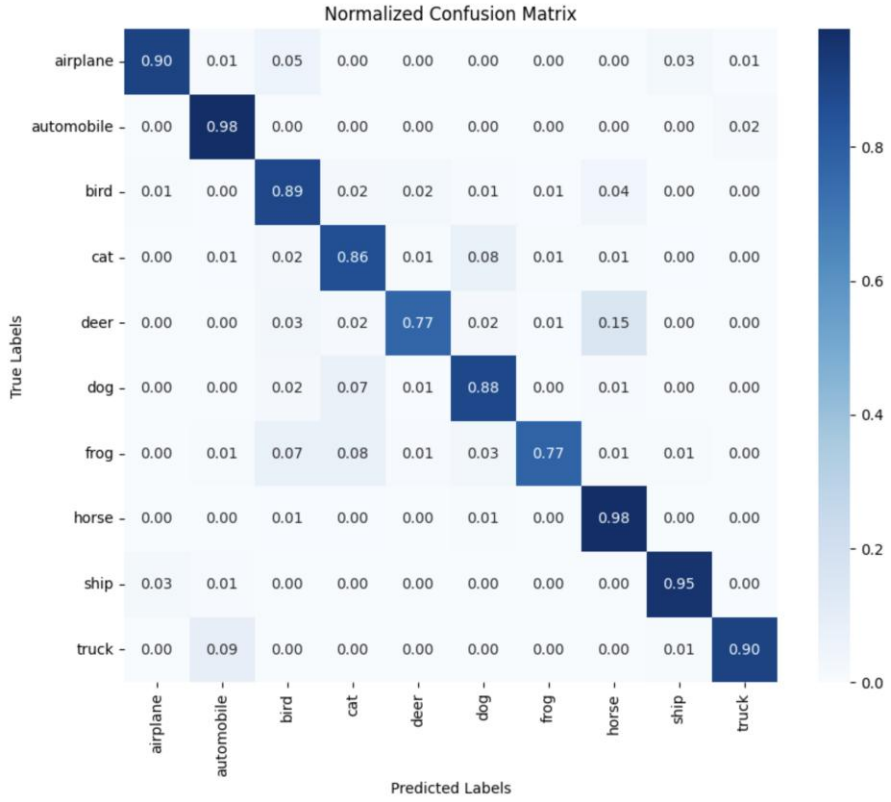

```
    1. Doberman, Doberman pinscher          | 91.20%
    2. miniature pinscher                   |  7.31%
    3. toy terrier                          |  0.36%
--------------------------------------------------
Classifying: kat_truck.jpeg
  Top-3 Predictions:
    1. pickup, pickup truck                 | 78.55%
    2. moving van                           |  6.96%
    3. minibus                              |  4.06%
--------------------------------------------------
Classifying: komodo_dragon.jpeg
  Top-3 Predictions:
    1. Komodo dragon, Komodo lizard, dragon lizard, giant lizard, Varanus komodoensis | 99.16%
    2. Gila monster, Heloderma suspectum    |  0.36%
    3. common iguana, iguana, Iguana iguana |  0.04%
--------------------------------------------------
Classifying: mountain.jpeg
  Top-3 Predictions:
    1. alp                                  | 92.77%
    2. valley, vale                         |  5.92%
    3. cliff, drop, drop-off                |  0.32%
--------------------------------------------------
Classifying: racecar.jpeg
  Top-3 Predictions:
    1. racer, race car, racing car          | 89.63%
```

ASSIGNMENT 2

- Introduction to Coding CLIP (Contrastive Language-Image Pre-training)
- CIFAR-10
- Zero-Shot Generalization

ASSIGNMENT 2 PROGRESS

- Loading Datasets and Setting up Labels
- Text and Image Embeddings
- Confusion Matrix




```python
# Collate function is basically a traffic controller to put the information into a usable format
def collate_fn(batch):
  images = [item[0] for item in batch]
  labels = [item[1] for item in batch]
  return images, torch.tensor(labels)

batch_size = 252

# Dataloader here used the collate_fn we made cause this is how we want the data to be formatted as
dataloader = DataLoader(test_dataset, batch_size=batch_size, shuffle=False, collate_fn=collate_fn)

all_preds = []
all_labels = []

# no_grad means no gradient calculations will be sued this way to save memory when just getting prediction
with torch.no_grad():

    for images, labels in dataloader:

        inputs = processor(images=images, return_tensors="pt").to(device)

        image_features = model.get_image_features(**inputs)

        image_features = image_features.pooler_output

        image_features = torch.nn.functional.normalize(image_features, p=2, dim=-1)

        similarity_scores = image_features @ text_features.T

        # this code instrcuts to make one final decision choosing the largest similarity scores from the m
        predictions = similarity_scores.argmax(-1)

        # these predictions are on the gpu cpu() switches it to gpu and numpy converts the tensor into an
        # unrolls the array and puts each prediction into the list
        all_preds.extend(predictions.cpu().numpy())

        all_labels.extend(labels.numpy())

# converting these large lists into numpy arrays more optimized
all_preds = np.array(all_preds)
all_labels = np.array(all_labels)
```