THE UNIVERSITY OF TEXAS AT DALLAS

Referring Expression Comprehension

Group 10
Cheung, Kevin Tak Hay (kxc220019)
Pingili, Nithin (nxp162430)
Xu, Jerry (zxx200003)

Introduction

- We trained several models to perform a visual-language task – Referring Expression Comprehension
  - Utilized a pre-trained model called FLAVA
  - Built an encoder-decoder model and a decoder-only model

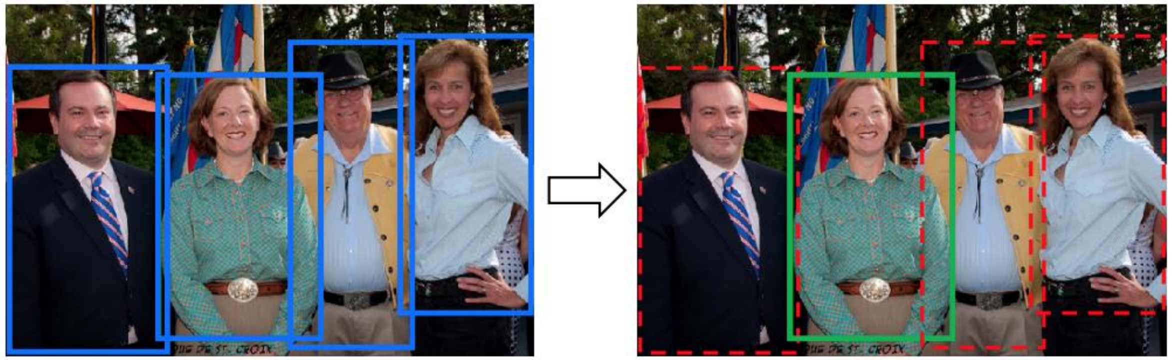
- Due to limited resources (GPUs), we cannot perform larger scale training
  - Limited the epoch and sample size to minimum

Task : Referring Expression Comprehension

- Given an image and a set of text captions, localizing a target object in the image described by the referring expression phrased in natural language

- REC is important for other vision-language tasks such as Visual Question Answering and Visual Dialogue

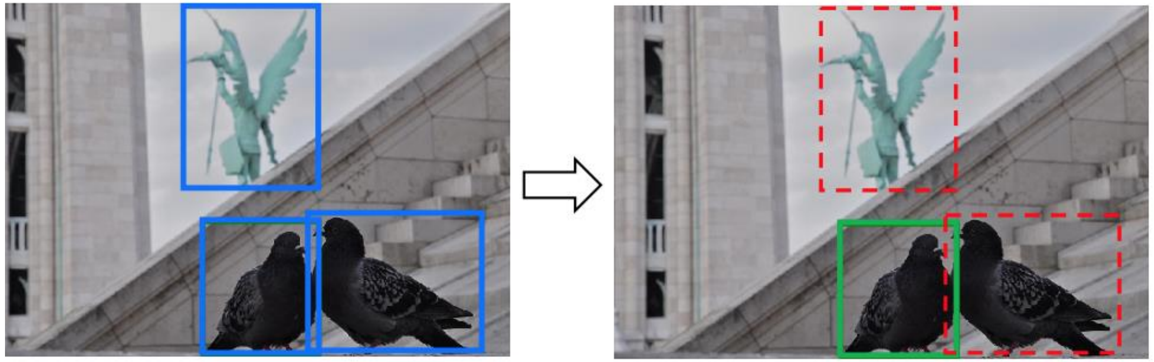
Expression: a lady standing next to a man wearing a blue suit and tie



Whole image and region proposals

Chosen region in green

Expression: bird directly below light blue statue



Whole image and region proposals

Chosen region in green

Data

- We used the RefCOCO family of datasets (RefCOCO, RefCOCO+, RefCOCOg). RefCOCO family are the annotations on top of MSCOCO (Common Objects in Context) dataset

- RefCOCO and RefCOCO+ expressions are strictly appearance-based descriptions (i.e., “person to the left” is an invalid description)

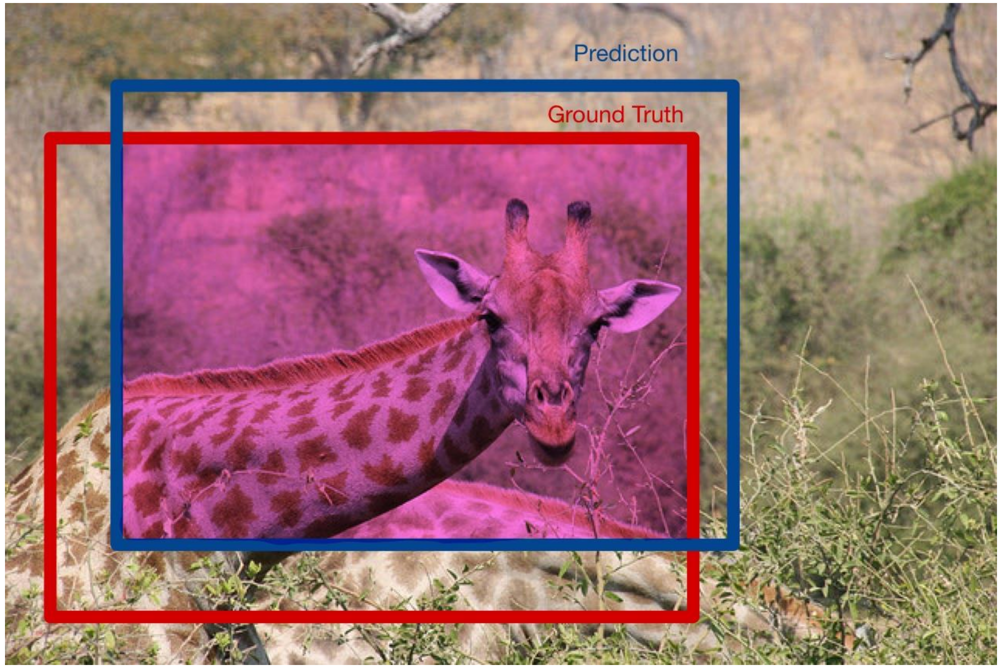
- Due to limited resources, we are only allowed to train on 50k examples from RefCOCO dataset
  - Very insufficient data size. Most VL models are trained on millions of samples

Evaluation Metric

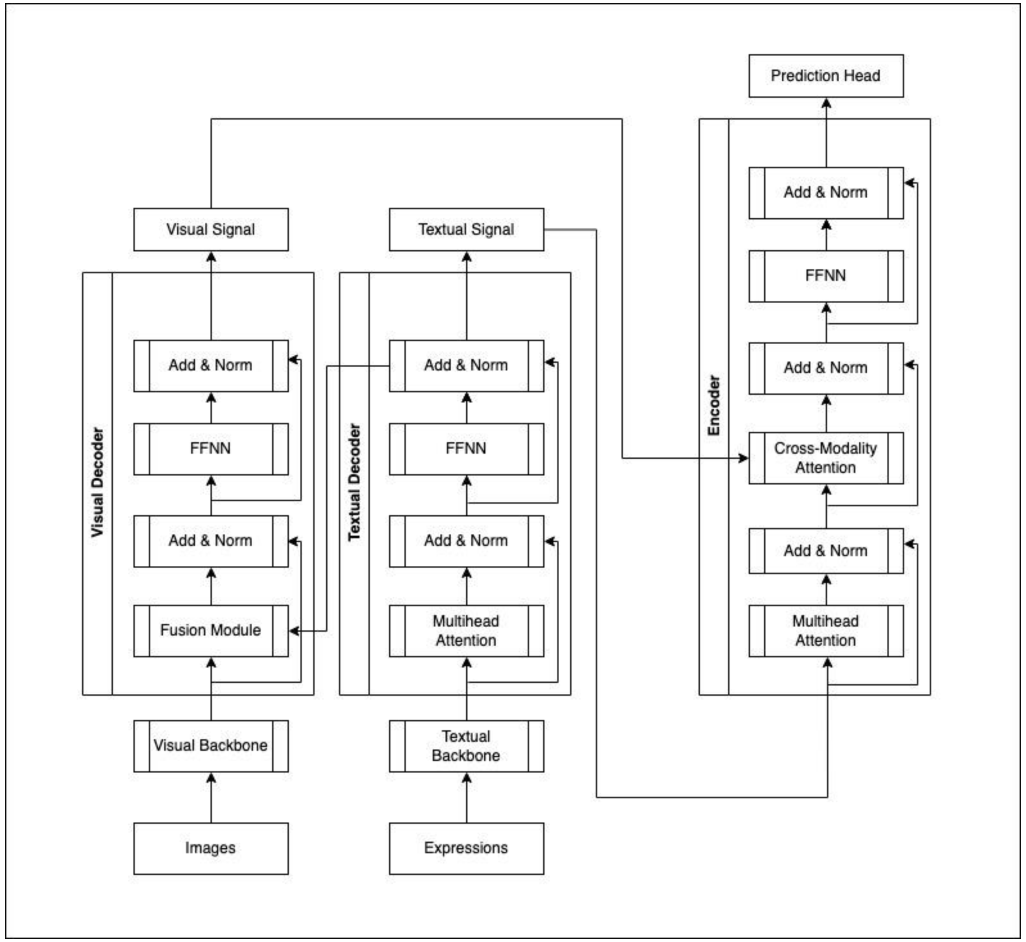
- Intersection over Union (IoU) is always used in object detection task

- It specifies the amount of overlap between the predicted and ground truth bounding box

- Widely used standard would be Acc@0.5 / Precision@0.5, which means correct classification if IoU > 0.5



1 – Encoder-Decoder Model - Introduction

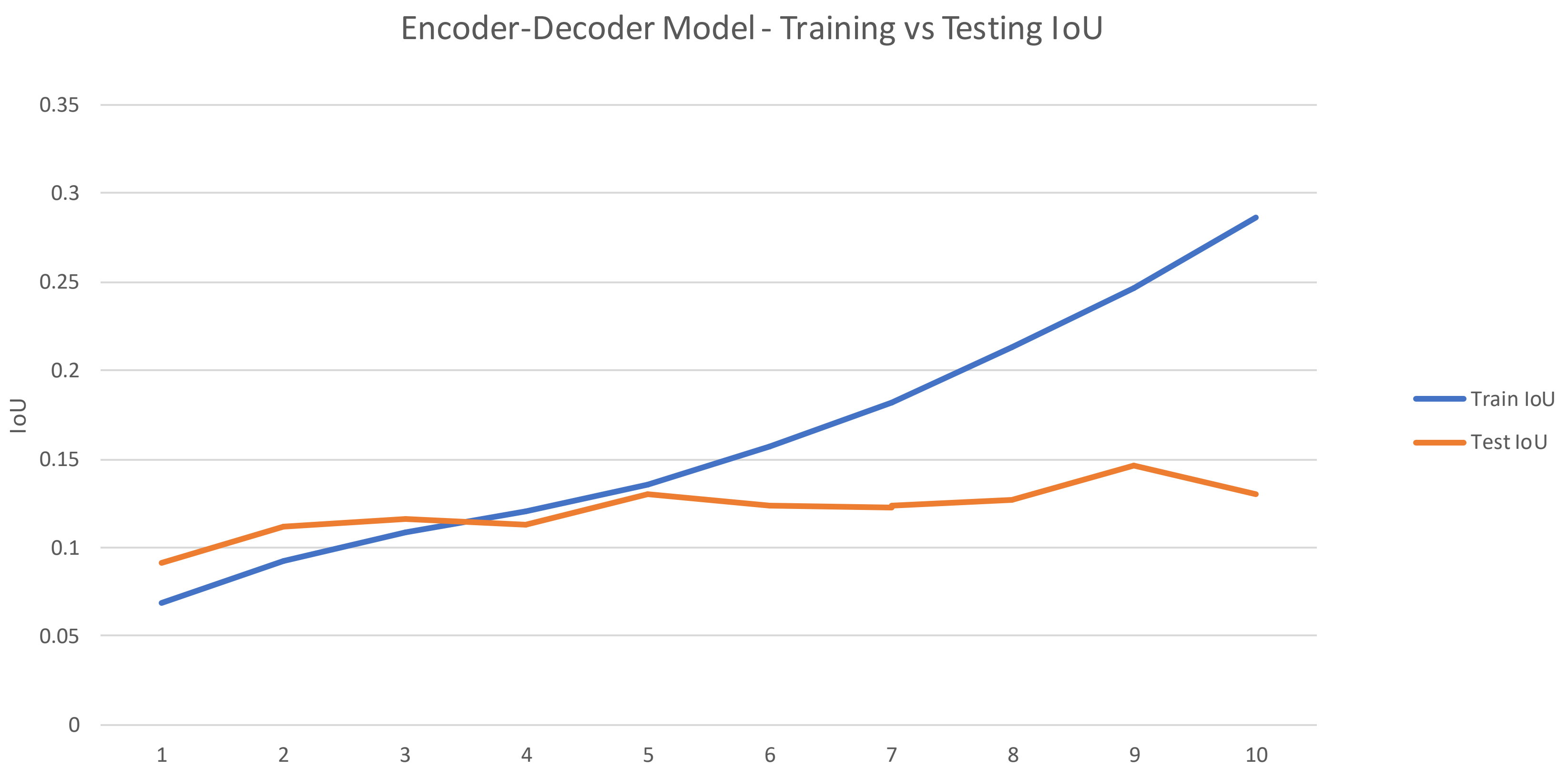


1 – Encoder-Decoder Model - Introduction

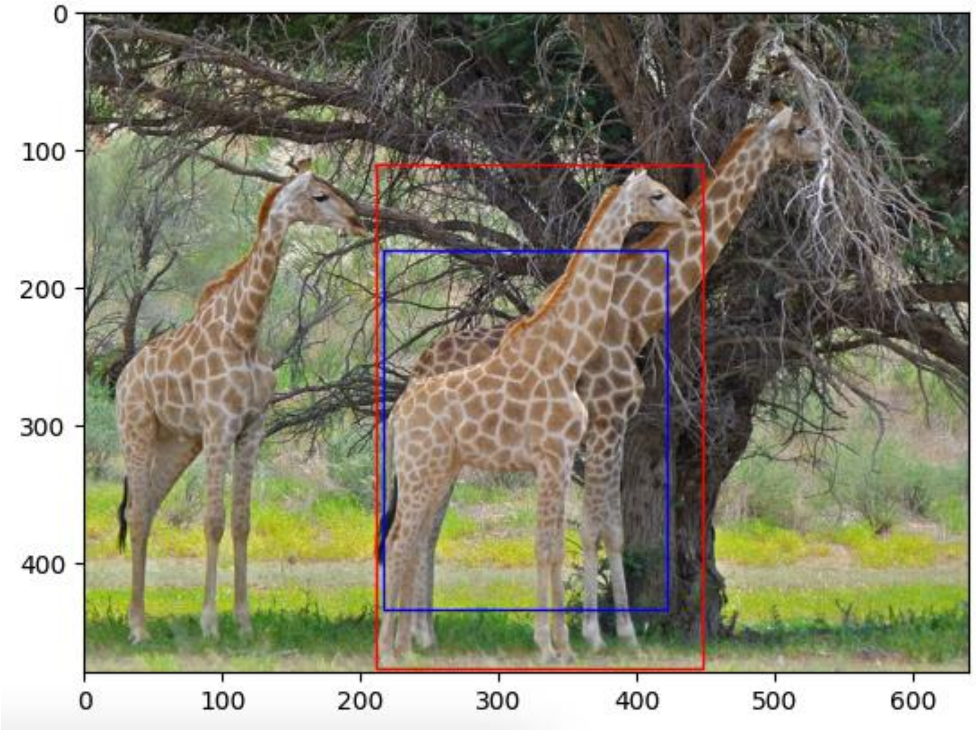
```python
class RECEncoder(nn.Module):

    def __init__(self, n = 2, nhead = 8, hidden_dim = 768):

        super().__init__()

        self.hidden_dim = hidden_dim
        self.nhead = nhead
        self.n = n
        self.text_encoder = AutoModel.from_pretrained('bert-base-uncased')
        self.visual_encoder = ResNetFeatureModel(output_layer='avgpool')
        self.image_hidden_size = 2048
        self.hidden_layer_1 = nn.Linear(self.image_hidden_size, 768)

        self.text_attentions = nn.ModuleList()
        self.text_FFNNs = nn.ModuleList()
        self.text_norms1 = nn.ModuleList()
        self.text_norms2 = nn.ModuleList()

        self.visual_attentions = nn.ModuleList()
        self.visual_FFNNs = nn.ModuleList()
        self.visual_norms1 = nn.ModuleList()
        self.visual_norms2 = nn.ModuleList()
        self.dropout = nn.Dropout(0.1)


        for i in range(self.n):

            self.text_attentions.append(nn.MultiheadAttention(self.hidden_dim, self.nhead, 0.1))
            self.text_norms1.append(nn.LayerNorm(self.hidden_dim))
            self.text_FFNNs.append(nn.Linear(self.hidden_dim, self.hidden_dim))
            self.text_norms2.append(nn.LayerNorm(self.hidden_dim))

            self.visual_attentions.append(nn.MultiheadAttention(self.hidden_dim, self.nhead, 0.1))
            self.visual_norms1.append(nn.LayerNorm(self.hidden_dim))
            self.visual_FFNNs.append(nn.Linear(self.hidden_dim, self.hidden_dim))
            self.visual_norms2.append(nn.LayerNorm(self.hidden_dim))



    def forward(self, image, expr):

        text_output = self.text_encoder(**expr)
        text_feature = text_output.last_hidden_state[:, 0, :]
        img_feature = self.hidden_layer_1(self.visual_encoder(image))

        for text_attention, tFFNN, tnorm1, tnorm2, visual_attention, vFFNN, vnorm1, vnorm2 in zip(self.text_attentions, self.text_FFNNs,

            attn_output, _ = text_attention(text_feature, text_feature, text_feature)
            attn_output = tnorm1(text_feature + self.dropout(attn_output))
            text_feature = tnorm2(self.dropout(tFFNN(attn_output)) + attn_output)


            visual_attn_output, _ = visual_attention(img_feature, text_feature, text_feature)
            visual_attn_output = vnorm1(img_feature + self.dropout(visual_attn_output))
            img_feature = vnorm2(self.dropout(vFFNN(visual_attn_output)) + visual_attn_output)



        return img_feature, text_feature
```

```python
class RECDecoder(nn.Module):

    def __init__(self, n = 2, nhead = 8, hidden_dim = 768):

        super().__init__()

        self.hidden_dim = hidden_dim
        self.nhead = nhead
        self.n = n


        self.attentions = nn.ModuleList()
        self.EDattentions = nn.ModuleList()
        self.FFNNs = nn.ModuleList()
        self.norms1 = nn.ModuleList()
        self.norms2 = nn.ModuleList()
        self.norms3 = nn.ModuleList()

        self.dropout = nn.Dropout(0.1)

        for i in range(self.n):

            self.attentions.append(nn.MultiheadAttention(self.hidden_dim, self.nhead, 0.1))
            self.EDattentions.append(nn.MultiheadAttention(self.hidden_dim, self.nhead, 0.1))
            self.norms1.append(nn.LayerNorm(self.hidden_dim))
            self.FFNNs.append(nn.Linear(self.hidden_dim, self.hidden_dim))
            self.norms2.append(nn.LayerNorm(self.hidden_dim))
            self.norms3.append(nn.LayerNorm(self.hidden_dim))



    def forward(self, image, expr):


        for attention, EDattention, FFNN, norm1, norm2, norm3 in zip(self.attentions, self.EDattentions, self.FFNNs,

            attn_output, _ = attention(expr, expr, expr)
            attn_output = norm1(expr + self.dropout(attn_output))
            EDattn_output, _ = EDattention(attn_output, image, image)
            EDattn_output = norm2(attn_output + self.dropout(EDattn_output))
            expr = norm3(self.dropout(FFNN(EDattn_output)) + EDattn_output)

        return expr
```

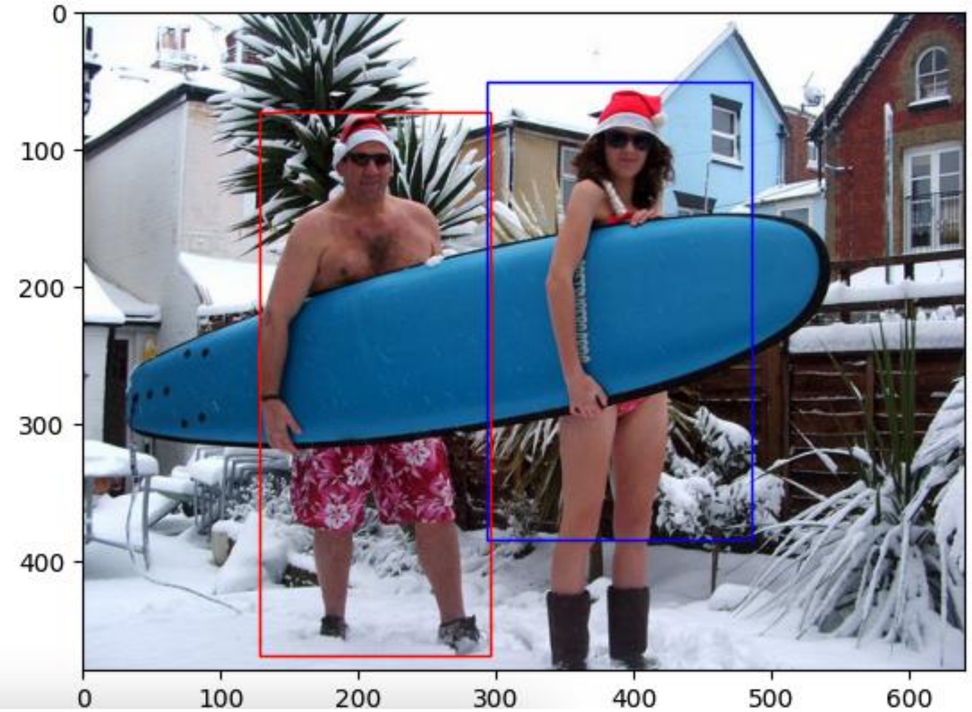
1 – Encoder-Decoder Model - Performance



1 – Encoder-Decoder Model - Examples

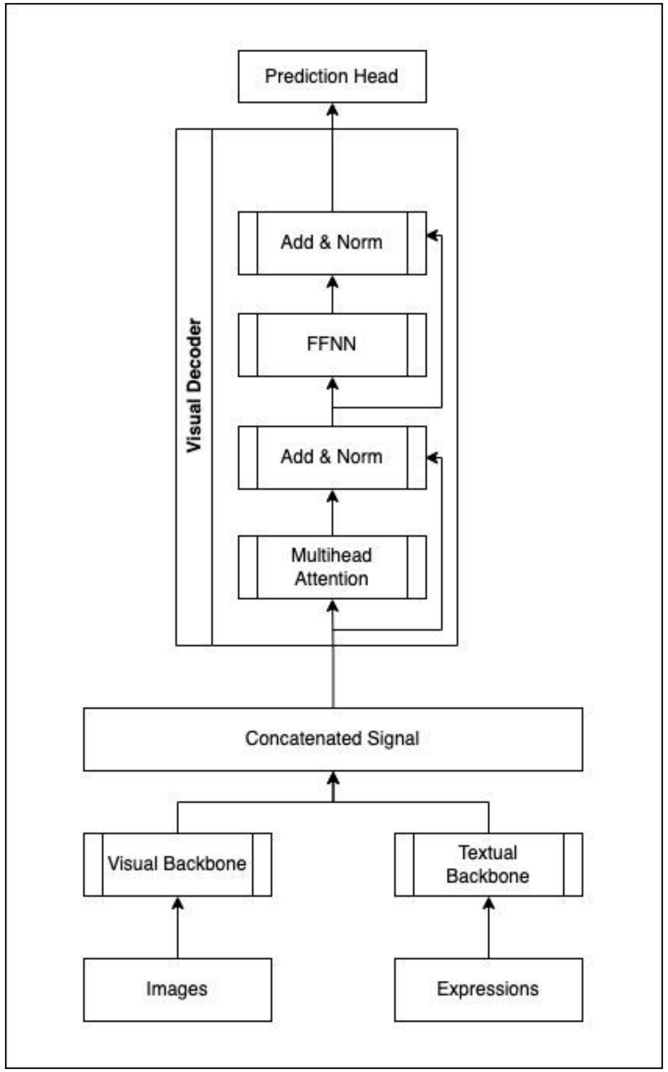


Expression: small giraffe in the middle first to us



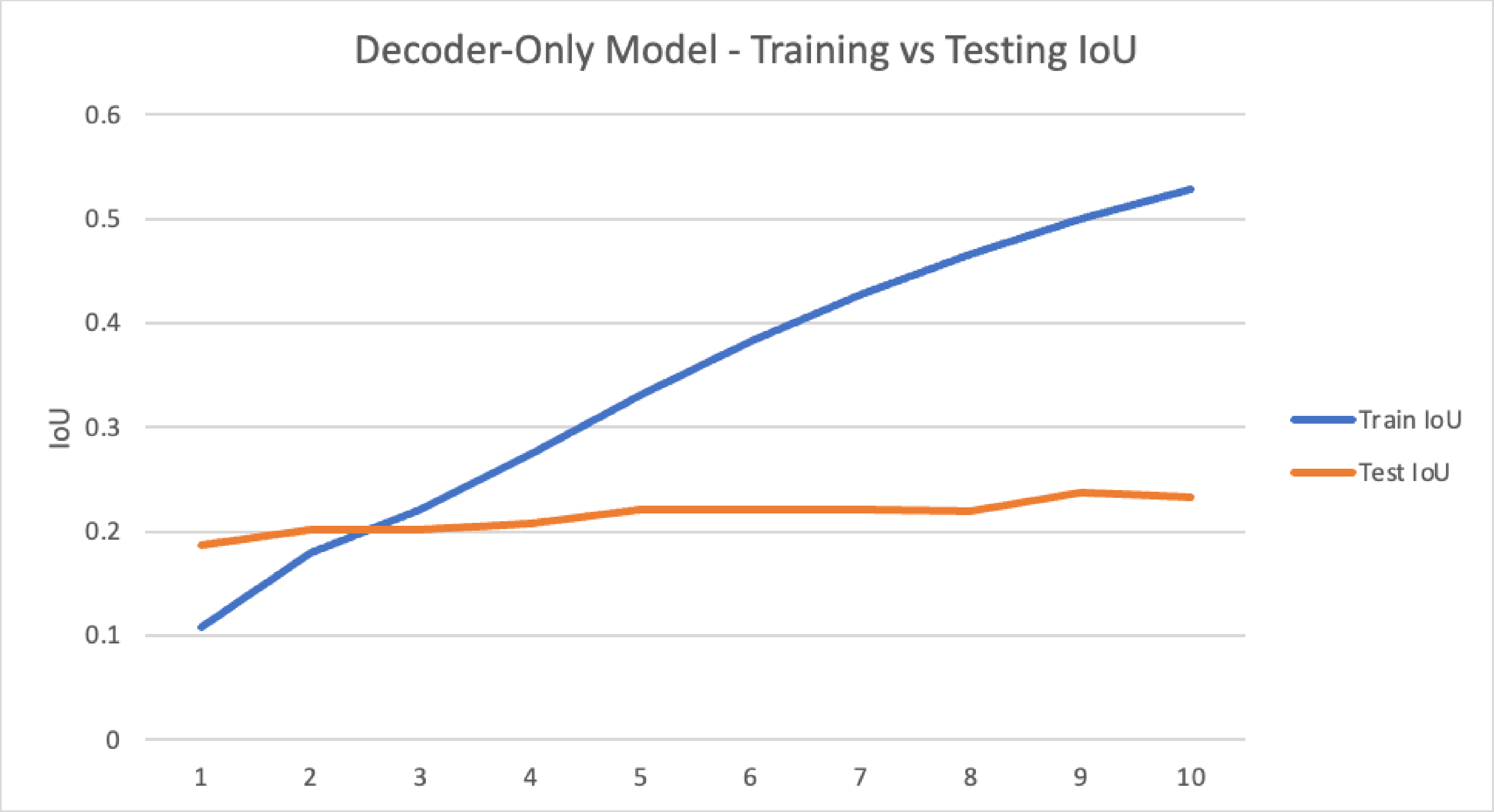
Expression: man in back of surfboard

2 – Decoder-Only Model - Introduction

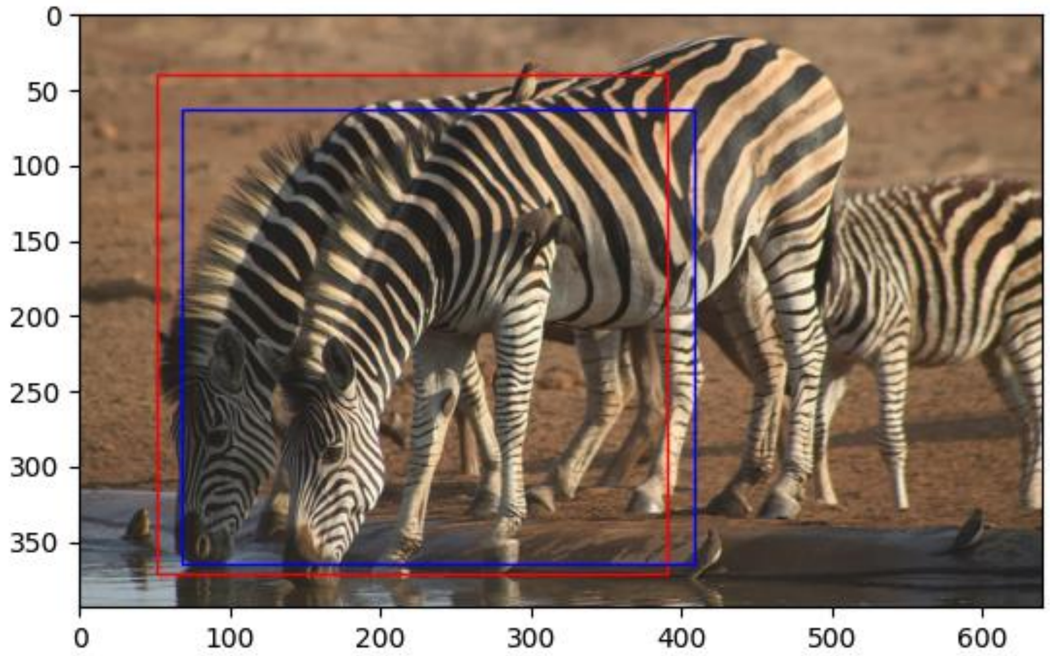



```python
class BertResNetModel(nn.Module):
    def __init__(self, text_pretrained='bert-base-uncased'):
        super().__init__()
        self.text_encoder = AutoModel.from_pretrained(text_pretrained)
        self.visual_encoder = ResNetFeatureModel(output_layer='avgpool')
        self.image_hidden_size = 2048

        self.hidden_layer_1 = nn.Linear(self.text_encoder.config.hidden_size + self.image_hidden_size, 512)

        self.attentions = nn.ModuleList()
        self.FFNNs = nn.ModuleList()
        self.norms = nn.ModuleList()


        for i in range(6):
            self.attentions.append(nn.MultiheadAttention(512, 8, 0.5))
            self.FFNNs.append(nn.Linear(512,512))
            self.norms.append(nn.LayerNorm(512))

        self.output_layer = nn.Linear(512, 4)
        self.dropout = nn.Dropout(0.5)
        self.act_1 = nn.ReLU()
        self.act_2 = nn.Sigmoid()

    def forward(self, text, image):
        text_output = self.text_encoder(**text)
        text_feature = text_output.last_hidden_state[:, 0, :]
        img_feature = self.visual_encoder(image)
        features = torch.cat((text_feature, img_feature), 1)

        x = self.act_1(self.hidden_layer_1(features))

        for attention, FFNN, norm in zip(self.attentions, self.FFNNs, self.norms):
            #x_norm = norm(x)
            attn_output, _ = attention(x, x, x)
            attn_output = self.dropout(attn_output)
            x = norm(x + attn_output)
            x = FFNN(x)
            #x = self.act_1(x)



        prediction_head = self.act_2(self.output_layer(x))

        return prediction_head
```

2–Decoder-Only Model - Performance



2–Decoder-Only Model - Examples

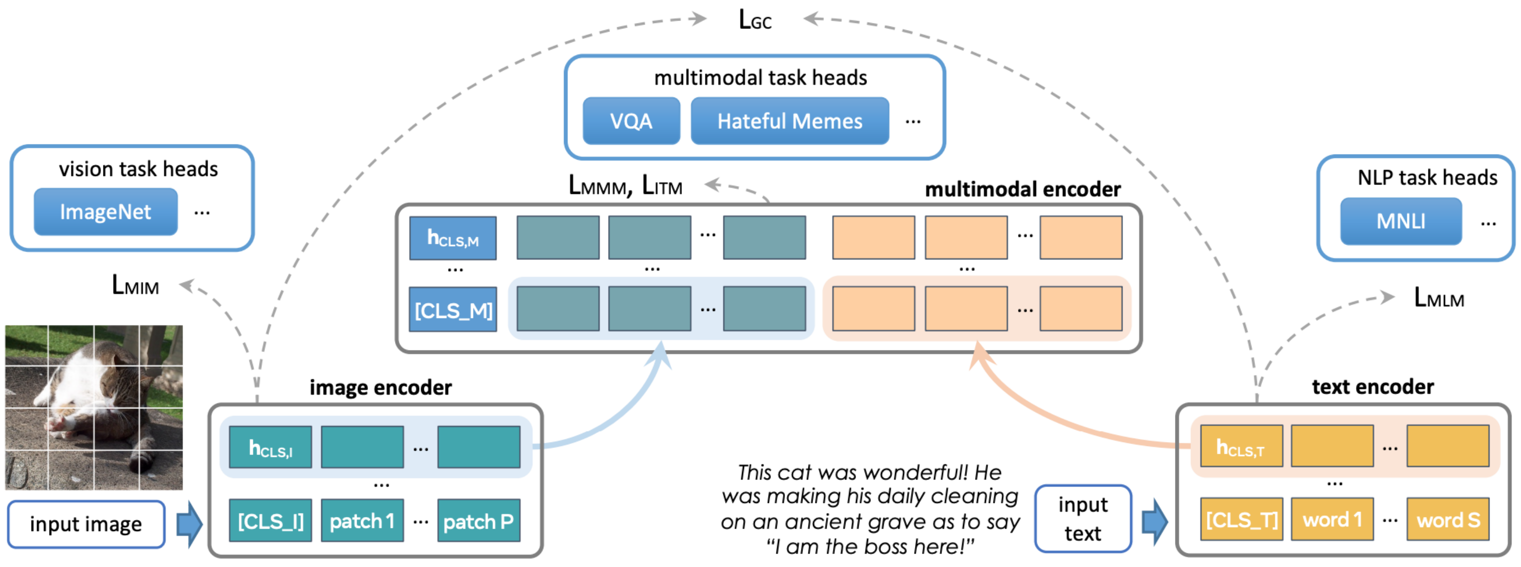


Expression: drinking zebra on the left

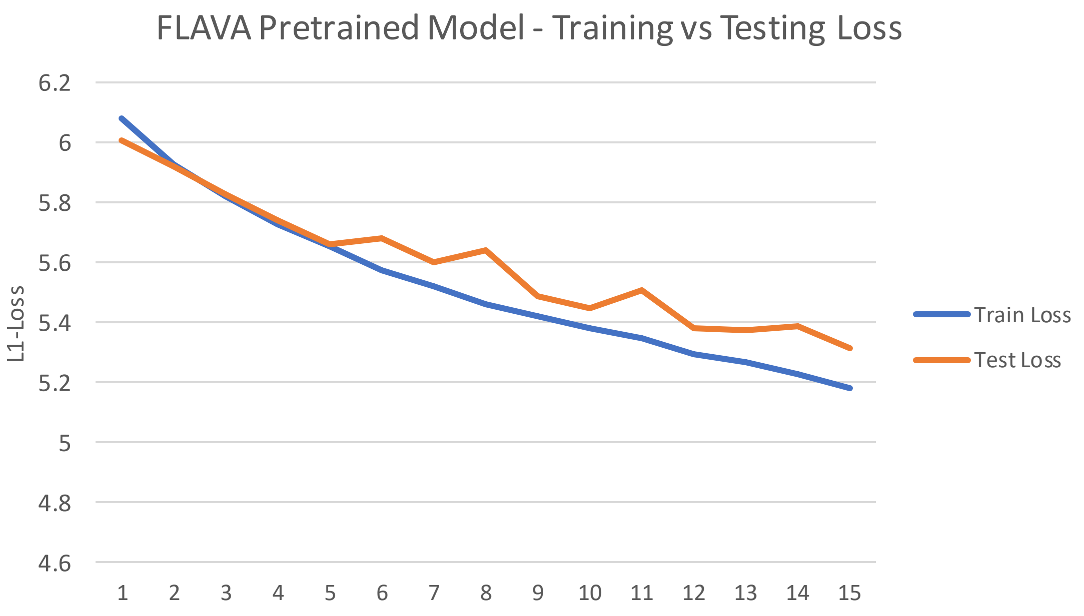
Expression: a man wearing black t shirt and holding a tennis ball in his hand

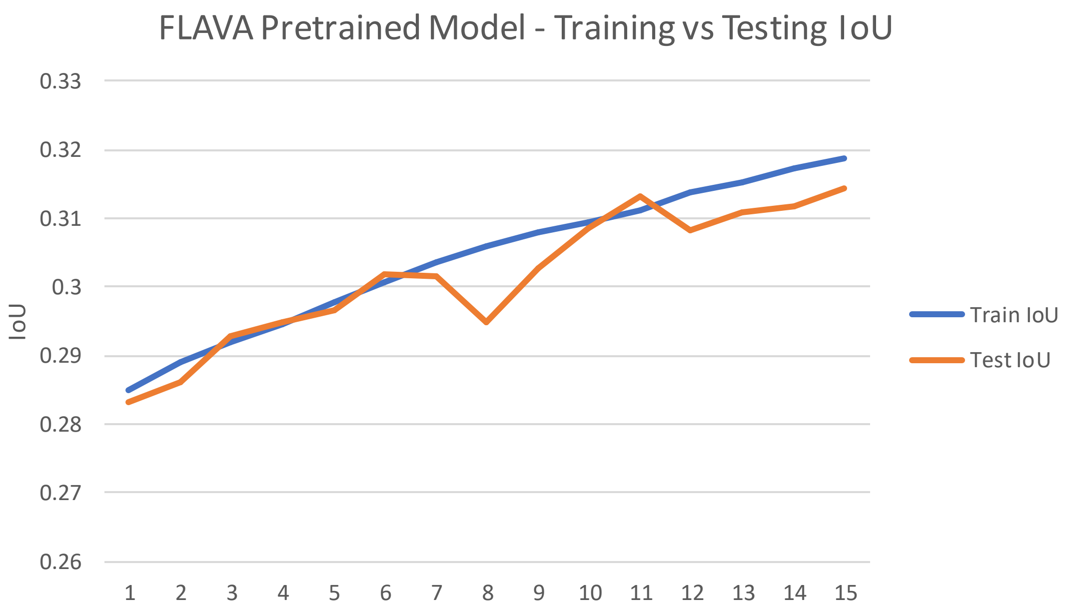
3 - Pre-trained FLAVA - Introduction

- Introduced by Singh et al in 2022; authors aimed to create a single universal model that is good at vision tasks, language tasks, and cross- and multi-modal vision and language tasks.
- FLAVA uses ViT to extract unimodal image representations, unimodal text representations, and fuse and align the image and text representations.
- During multimodal pretraining, authors trained for 46000 epochs.
- We trained our own prediction head for the REC task

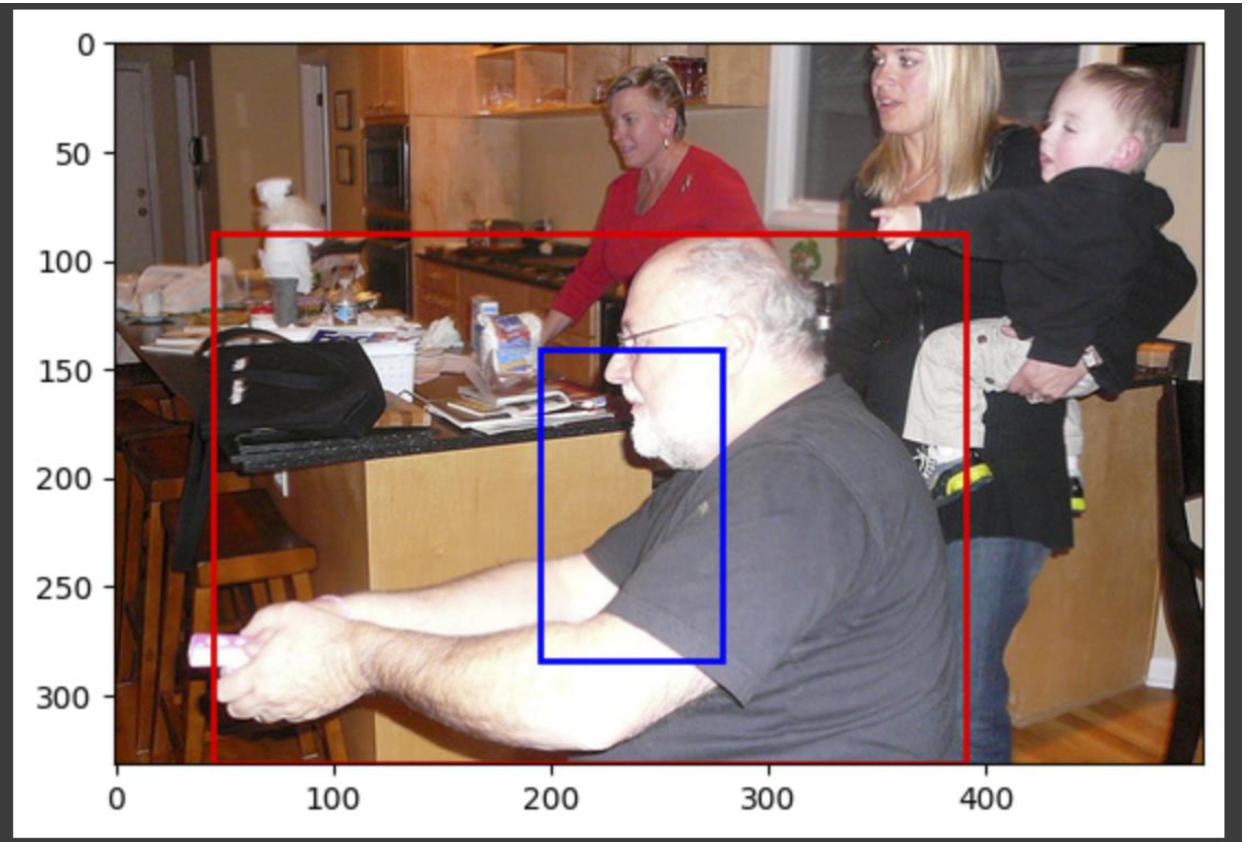


3- Pre-trained FLAVA - Performance





3 - Pre-training FLAVA – Examples



a man sitting down with two hands on a remote

the whole pizza with fresh greens

Improvements

- Train on more data
  - Pre-trained models requires more epoch (at least hundreds) and data for fine-tuning
  - Non-pretrained models with specific task still requires at least millions of image-text pairs

- Get better GPU access
  - The current computation cost is too high, and we cannot perform larger scale training, which results in a subpar performance

- Develop a better approach for multimodality fusion
  - Currently the fusion is done using multiheaded attentions. There are more ways to broadcast the text query signals to the images so that the multimodal inputs can be aligned better