Packages Set up

```
!pip install bertviz
!pip install transformers
```

```
import os, re
import time
import numpy as np
import pandas as pd

import io
from io import BytesIO
import torch
import torch.nn as nn
import torch.nn.functional as F
from torch.cuda.amp import autocast, GradScaler
from torch.utils.data import TensorDataset, random_split, DataLoader, RandomSampler
from transformers import T5Tokenizer, T5ForConditionalGeneration
from transformers import AdamW, get_linear_schedule_with_warmup
```

GCP connectivity and Data set up

```
! ls -lrt /content/drive/MyDrive/*.json
```

```
import os
os.environ['GOOGLE_APPLICATION_CREDENTIALS'] = '/content/drive/MyDrive/pacific-castle-
!echo $GOOGLE_APPLICATION_CREDENTIALS
```

```
from google.cloud import storage

storage_client = storage.Client()

buckets = storage_client.list_buckets()

print('-- List of buckets in project \"' + storage_client.project + '\"')

for b in buckets:
  print(b.name)
```

```
#Initialize google storage
#storage_client = storage.Client.from_service_account_json('pacific-castle-360400-a3ca
```

```
#Print·buckets·available
for·bucket·in·storage_client.list_buckets():
··print(bucket)

#Assign·bucket·name·being·used
bucket_name·=·'266csffile'
#bucket_name·=·'w266liwc'

#Get·bucket
bucket·=·storage_client.get_bucket(bucket_name)

#·#Show·list·of·files·in·bucket·and·list·the·files
#·filelist·=·list(bucket.list_blobs(prefix=''))
#·for·name·in·filelist:
#···print(name.name)


def read_parquet_google_cloud(file):
    '''This function reads a file from the google cloud storage bucket. Input
    parameters include the filename, encoding and CSV file separators.'''

    #Load Google Cloud storage client using service key

    blob = bucket.blob(file)
    read_back = bucket.blob(file)
    string_read_back = read_back.download_as_string()
    new_df = pd.read_parquet(io.BytesIO(string_read_back))

    return new_df


#Create dictonary to transform MBTI type into multiclass value from 0 to 15
valid_MBTI = {'ISTJ': 0, 'INTJ': 1, 'ESTJ': 2, 'ENTJ': 3, 'ENTP': 4, 'INTP': 5, \
              'ISTP': 6, 'ESTP': 7, 'ISFJ': 8, 'INFJ': 9, 'ESFJ': 10, \
              'ENFJ': 11, 'ENFP': 12, 'INFP': 13, 'ISFP': 14, 'ESFP': 15}


from bertviz import model_view
from transformers import T5Tokenizer, T5ForConditionalGeneration
t5_tokenizer = T5Tokenizer.from_pretrained('t5-small')
model = T5ForConditionalGeneration.from_pretrained('t5-small')
optimizer = torch.optim.AdamW(model.parameters(),
                  lr = 3e-5
                  )


def load_model_from_checkpoints( model, optimizer, path_to_checkpoint ):

    # load state from file
    checkpoint = torch.load('/content/drive/My Drive/W266/MBTI/model_checkpoints/t5-c]
    model.load_state_dict(checkpoint['model_state_dict'])
```

```
    optimizer.load_state_dict(checkpoint['optimizer_state_dict'])
    epoch = checkpoint['epoch']
    loss = checkpoint['loss']

    return model, optimizer, epoch, loss

model, optimizer, epoch, loss = load_model_from_checkpoints( model, optimizer, '/conte


#max_length=512
max_length = 64
# Model is utilizing entire 512 capacity from T5, but bertviz has scaling issues with
# so for the purposes of this notebook setting max length at 64


#load test data and labels files
test_df = read_parquet_google_cloud('t5_test_uniform.parquet')
test_mbti_labels = read_parquet_google_cloud('test_mbti_labels.parquet')
target_text_list = [ label + " " for label in test_mbti_labels['MBTI Type']]
test_df['target_text'] = target_text_list
```

Sample display for correct and wrong predictions

Accurate Prediction Sample

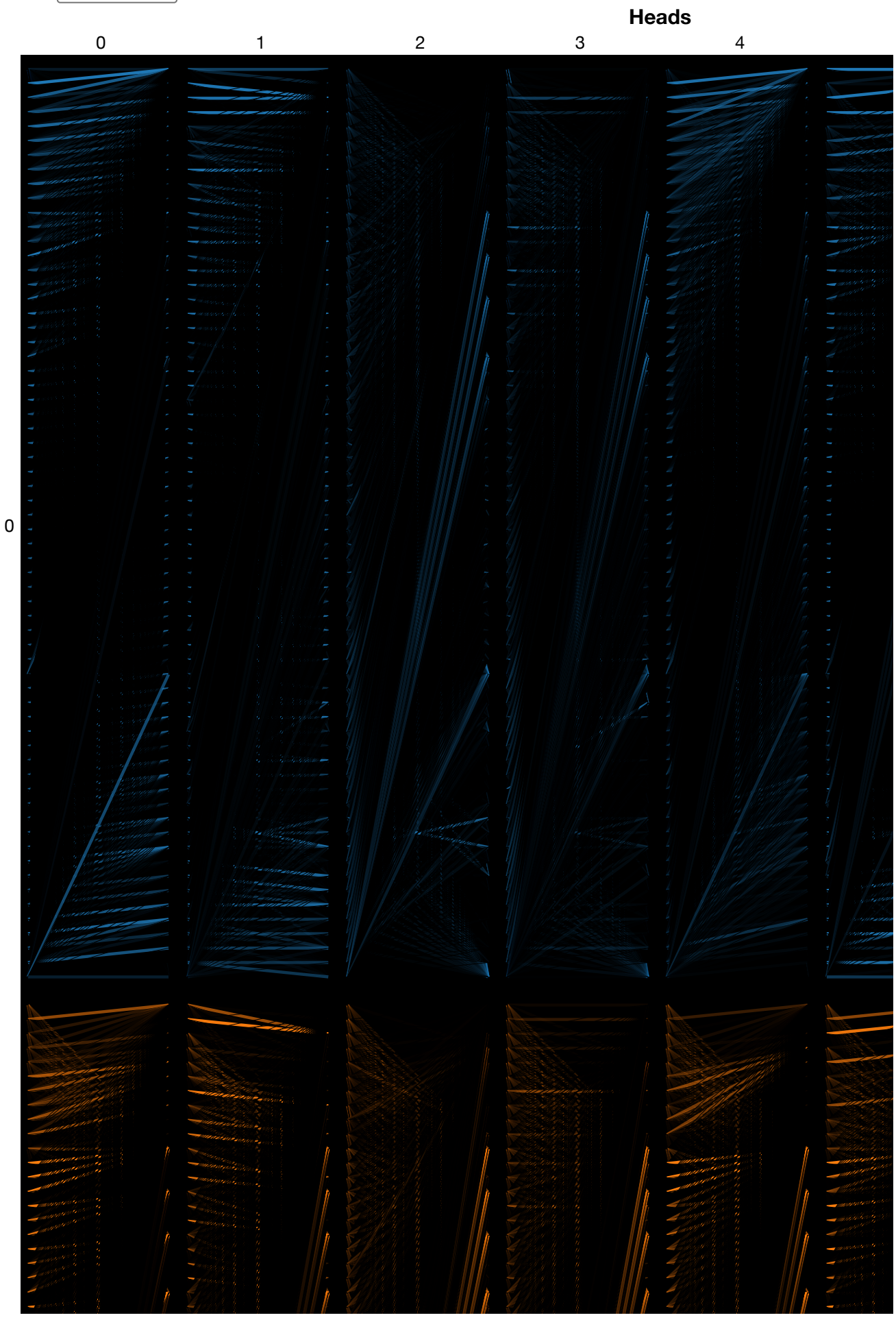
```
test_df[3:4]
```

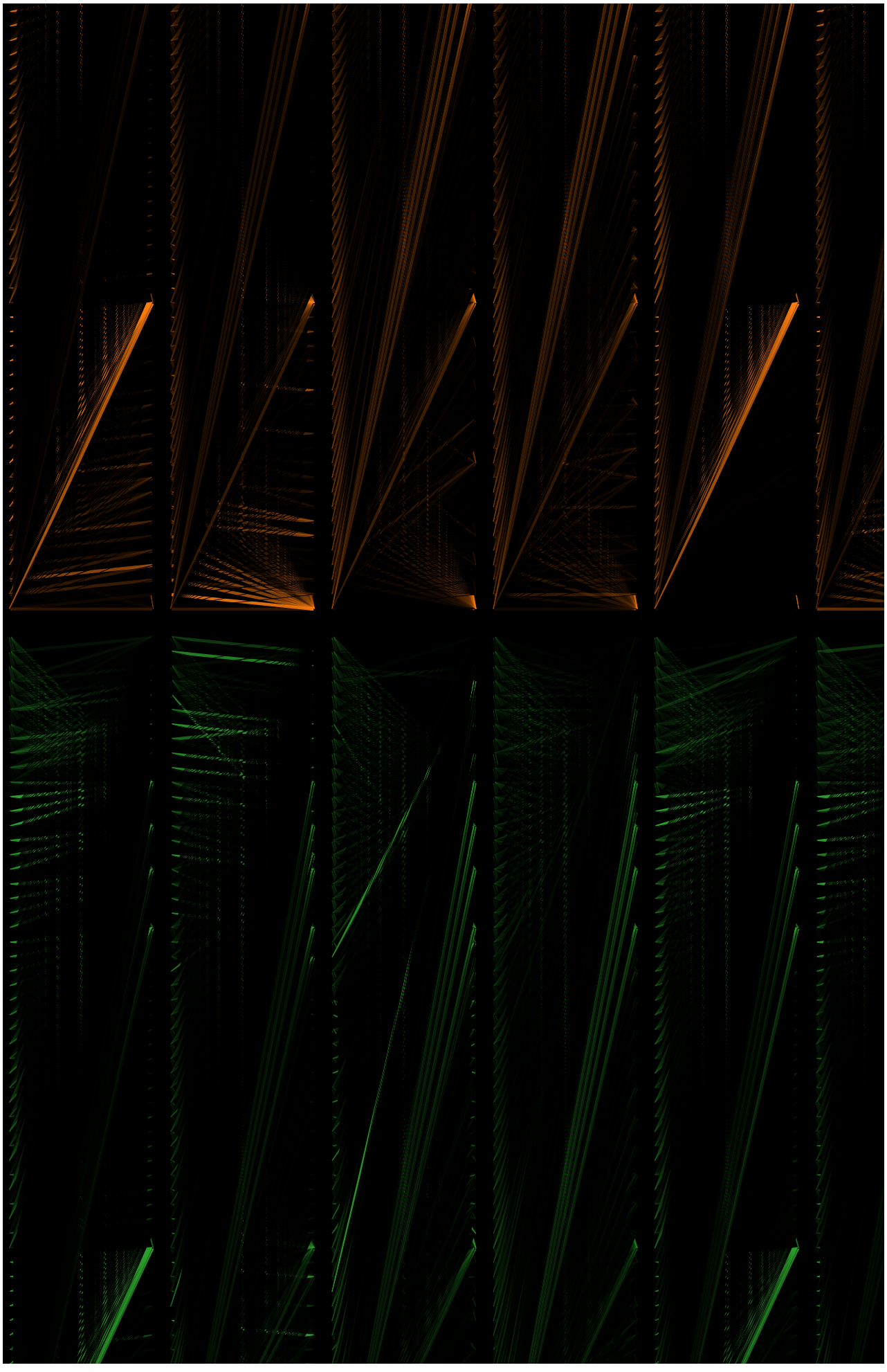
	Username	Age	Posts	Occupation	message	is_I	is_S	is_T	i
1400003	eric b	56.0	3620	subway motorman	^ i think so too!! tyrion and brienne are both...	True	False	True	F

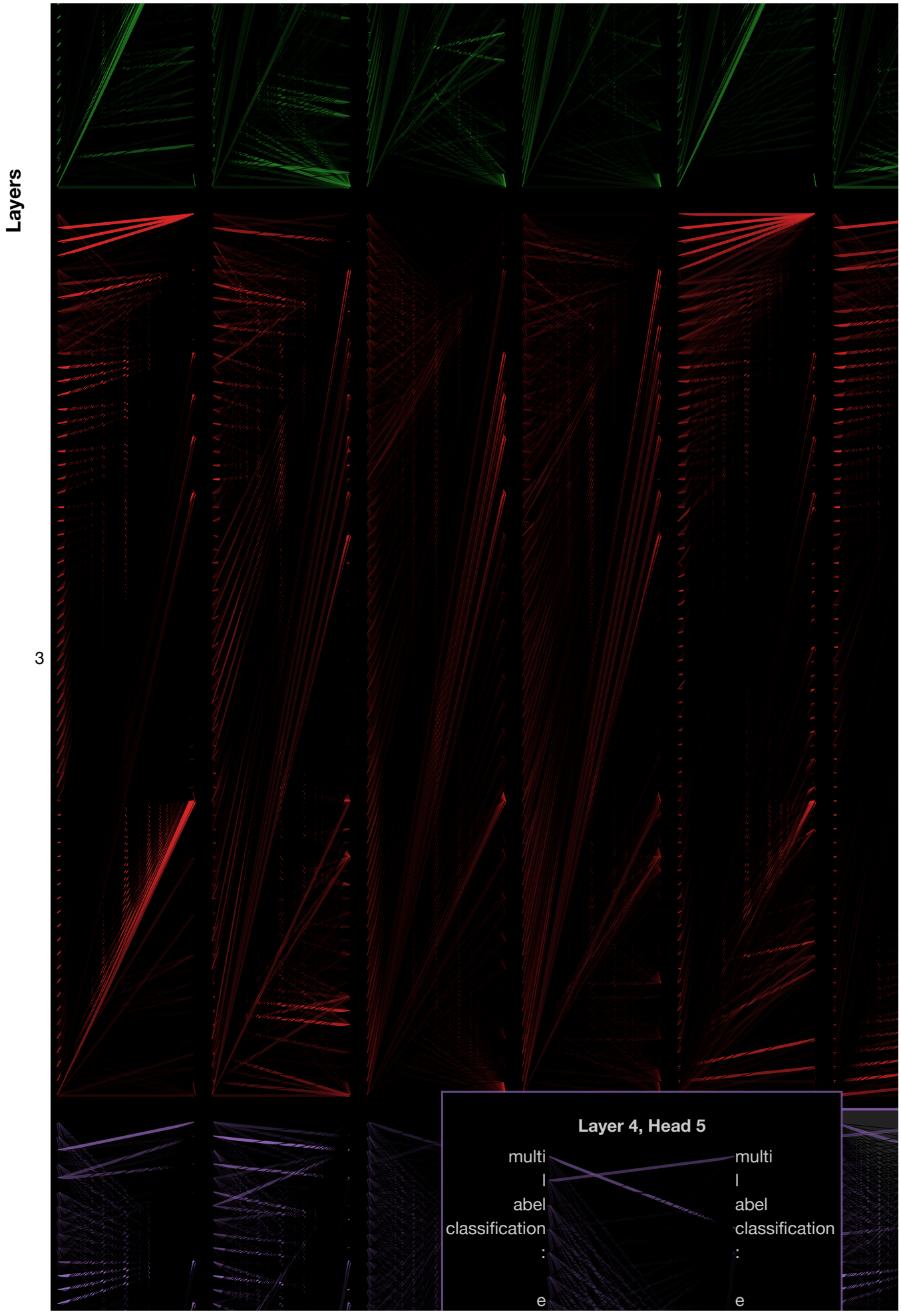
```
sample_correct_input1= test_df[3:4]['combined'].tolist()
sample_correct_target1= test_df[3:4]['target_text'].tolist()

c_encoder_input_ids = t5_tokenizer(sample_correct_input1, add_special_tokens=True, # a
                                   truncation=True,  # truncate longer inputs
                                   max_length=max_length,  # set max_length
                                   padding = 'max_length',  # add padding
                                   return_attention_mask=True,  # create attr
                                   return_tensors='pt').input_ids
```

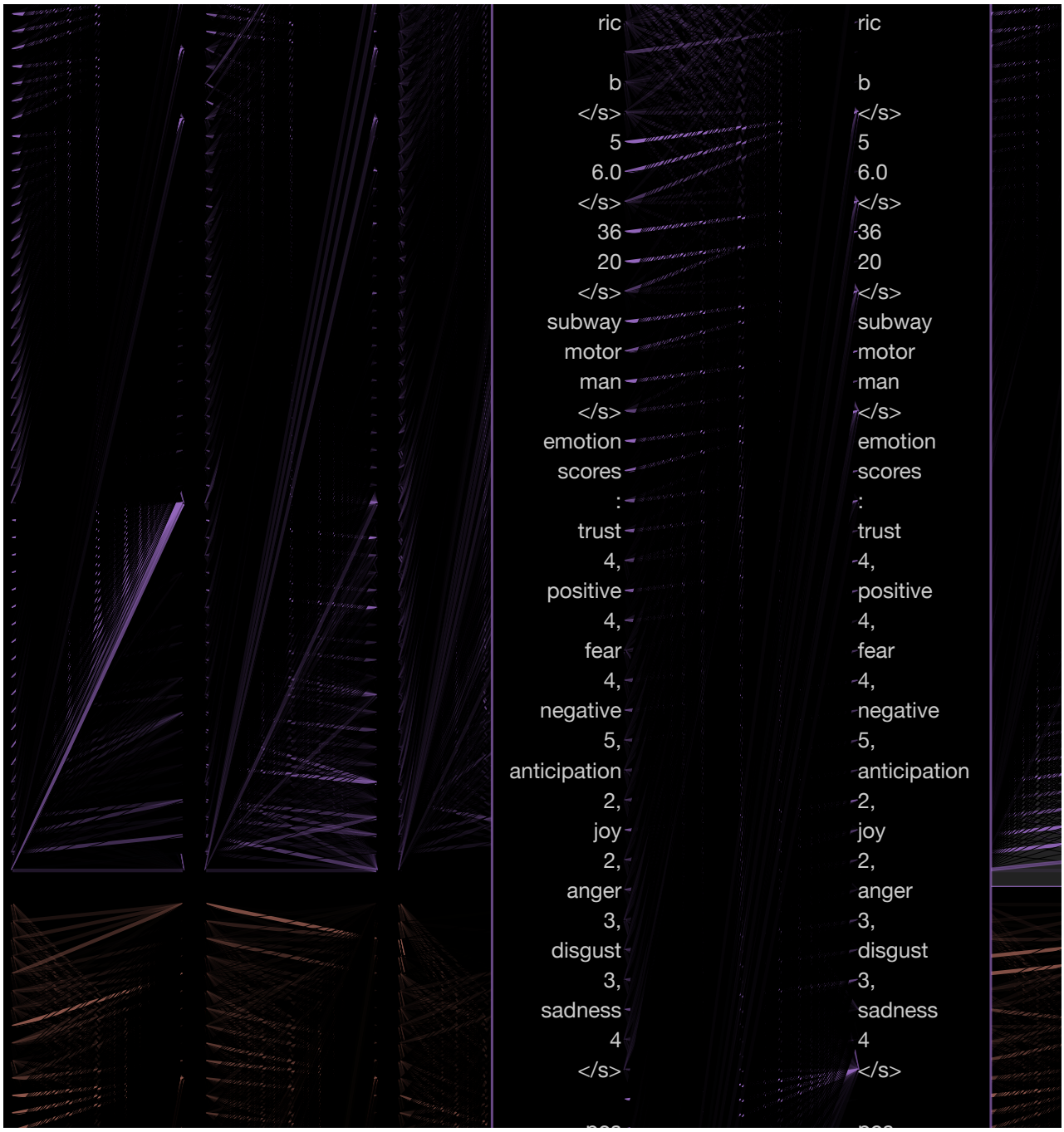
```
c_decoder_input_ids = t5_tokenizer(sample_correct_target1, add_special_tokens=True, #
                                                    truncation=True,  # truncate longer inputs
                                                    max_length=max_length,  # set max_length
                                                    padding = 'max_length',  # add padding
                                                    return_attention_mask=True,  # create attr
                                                    return_tensors='pt').input_ids


c_outputs = model(input_ids=c_encoder_input_ids, decoder_input_ids=c_decoder_input_ids

c_encoder_text = t5_tokenizer.convert_ids_to_tokens(c_encoder_input_ids[0])
c_decoder_text = t5_tokenizer.convert_ids_to_tokens(c_decoder_input_ids[0])

#call_html()

model_view(
    encoder_attention=c_outputs.encoder_attentions,
    decoder_attention=c_outputs.decoder_attentions,
    cross_attention=c_outputs.cross_attentions,
    encoder_tokens= c_encoder_text,
    decoder_tokens=c_decoder_text
)
```

Attention: Encoder




1
2


Layers
3
Layer 4, Head 5
multi
l
abel
classification
:
e
multi
l
abel
classification
:
e

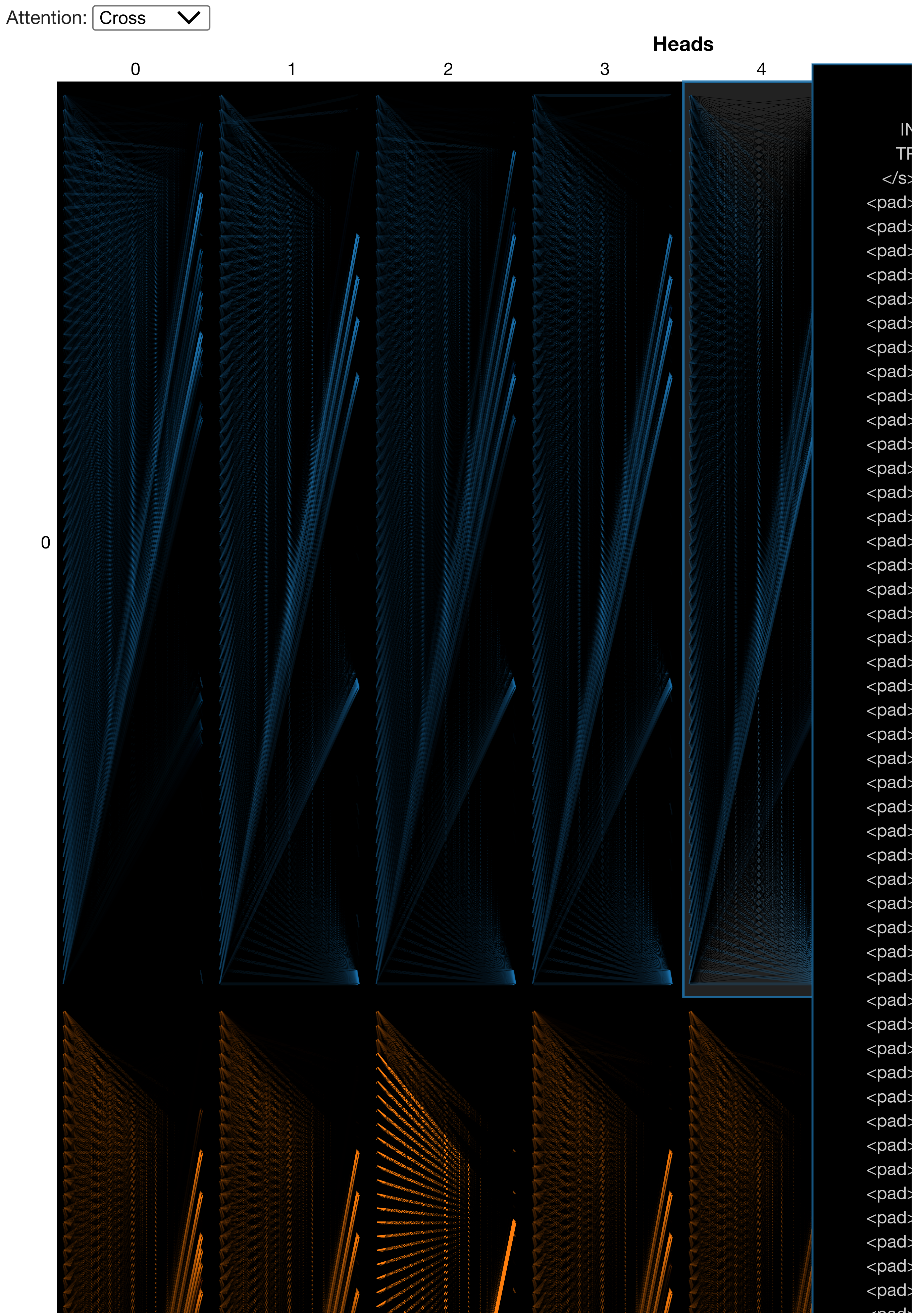


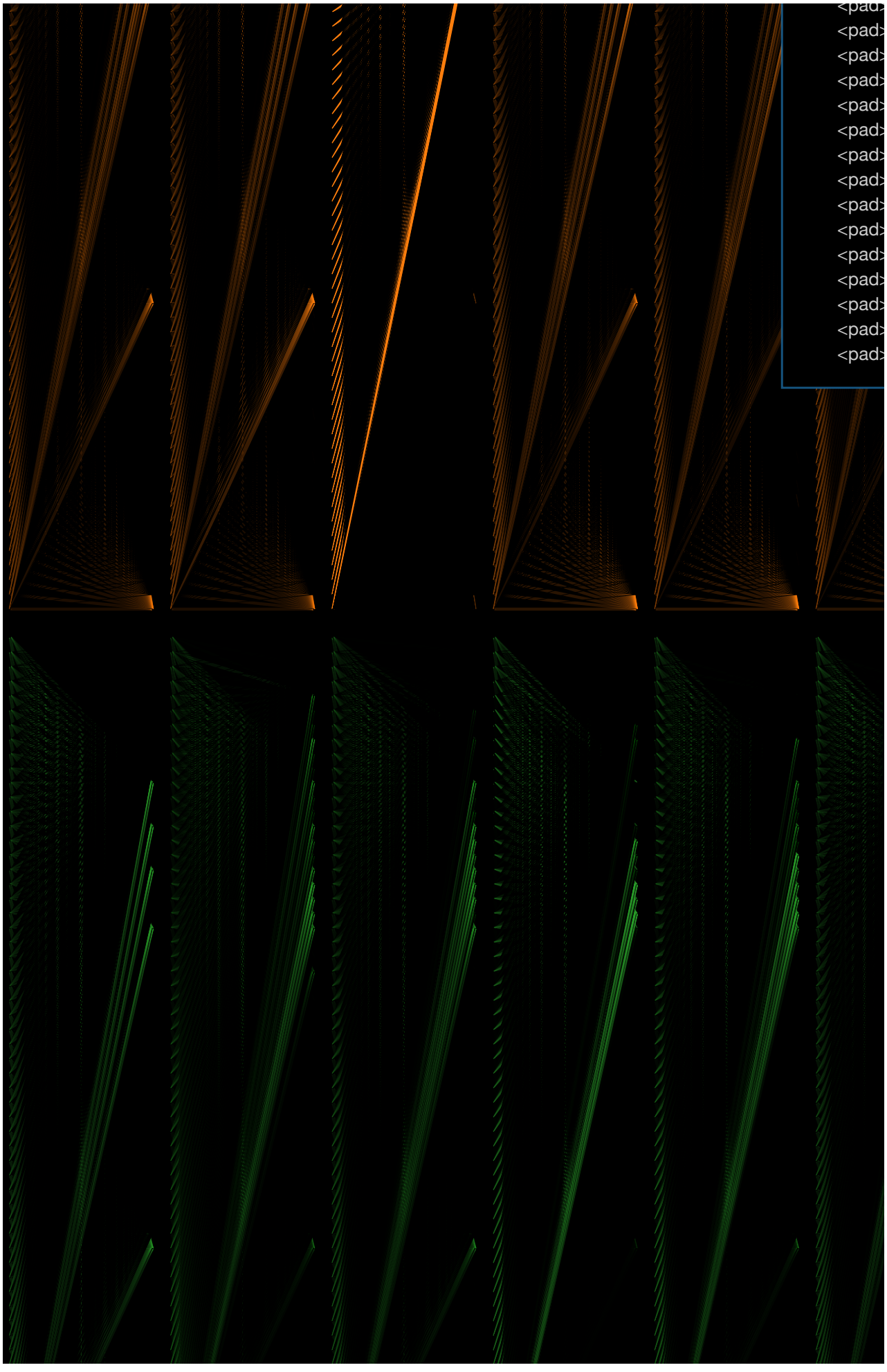
Inaccurate Prediction Sample

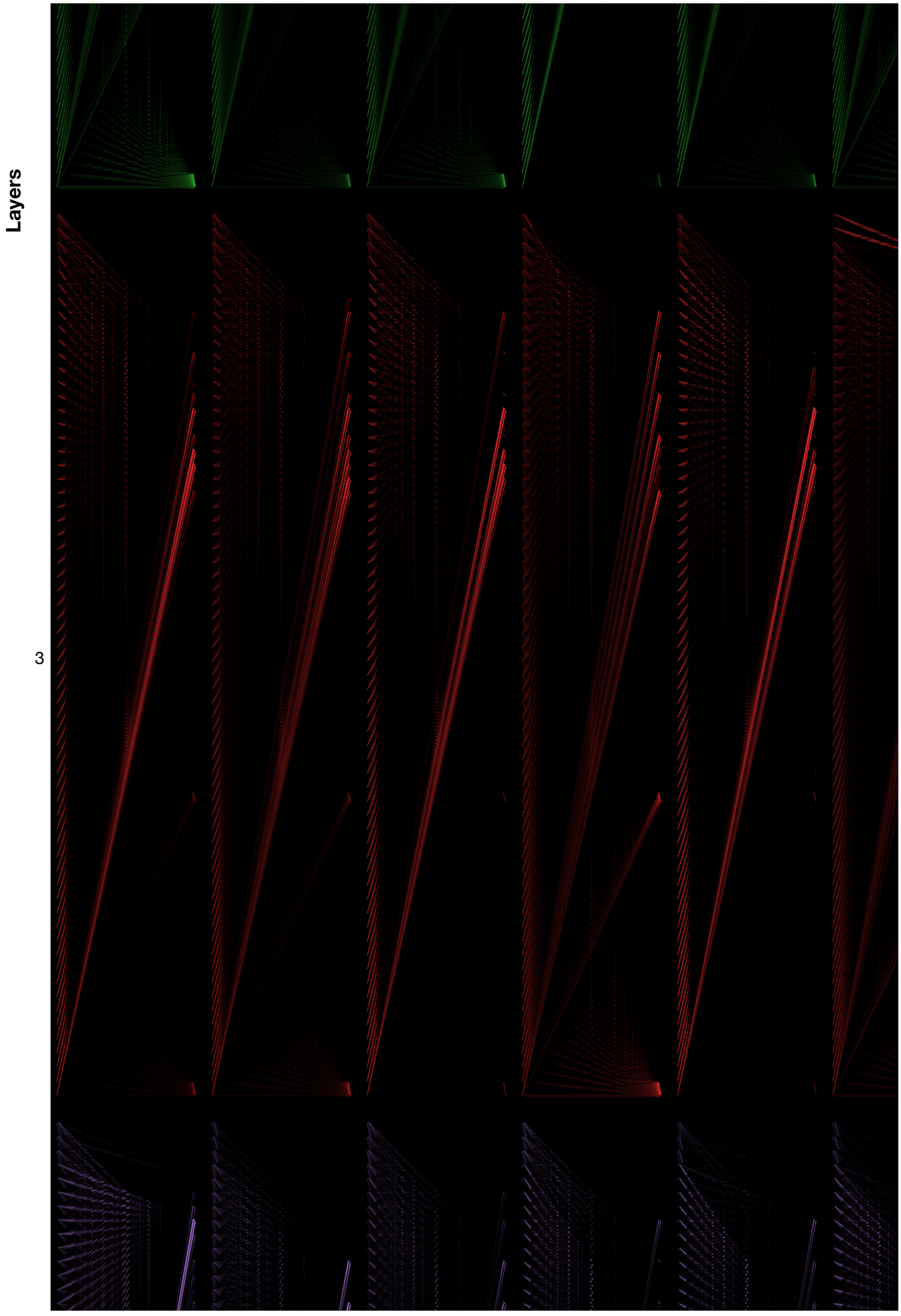
```
test_df[146:147]
```

	Username	Age	Posts	Occupation	message	is_I	is_S	is_T	i
1400146	evastover	23.0	77	freelance pianist	that is too freudian and slowmoving for my t...	True	False	False	F

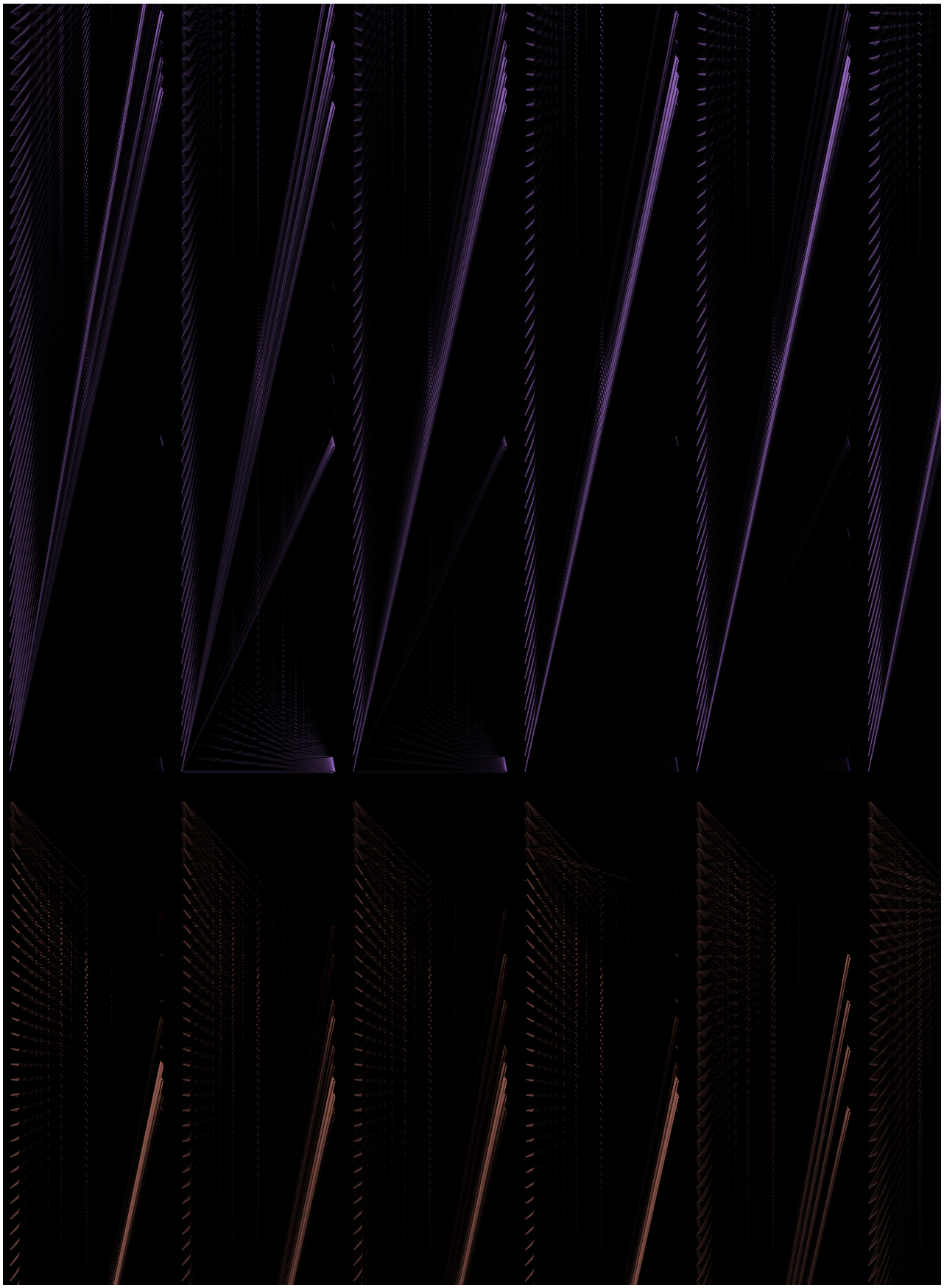
```
sample_wrong_input1= test_df[146:147]['combined'].tolist()
sample_wrong_target1= test_df[146:147]['target_text'].tolist()

w_encoder_input_ids = t5_tokenizer(sample_correct_input1, add_special_tokens=True, # a
                                                    truncation=True,  # truncate longer inputs
                                                    max_length=max_length,  # set max_length
                                                    padding = 'max_length',  # add padding
                                                    return_attention_mask=True,  # create attr
                                                    return_tensors='pt').input_ids

w_decoder_input_ids = t5_tokenizer(sample_correct_target1, add_special_tokens=True, #
                                                    truncation=True,  # truncate longer inputs
                                                    max_length=max_length,  # set max_length
                                                    padding = 'max_length',  # add padding
                                                    return_attention_mask=True,  # create attr
                                                    return_tensors='pt').input_ids


w_outputs = model(input_ids=w_encoder_input_ids, decoder_input_ids=w_decoder_input_ids

w_encoder_text = t5_tokenizer.convert_ids_to_tokens(w_encoder_input_ids[0])
w_decoder_text = t5_tokenizer.convert_ids_to_tokens(w_decoder_input_ids[0])

#call_html()

model_view(
    encoder_attention=w_outputs.encoder_attentions,
    decoder_attention=w_outputs.decoder_attentions,
    cross_attention=w_outputs.cross_attentions,
    encoder_tokens= w_encoder_text,
    decoder_tokens=w_decoder_text
)
```


Attention: Cross
Heads
0
1
2
3
4
0


1
2
<pad>
<pad>
<pad>
<pad>
<pad>
<pad>
<pad>
<pad>
<pad>
<pad>
<pad>
<pad>
<pad>
<pad>


Layers
3

4



Project team utilized BertViz module to get hands on experience with attention visualization as part of the MBTI

project. The bertviz module struggeld with max length > 64. We showcased one example of ACCURATE and INACCURATE prediction from the model. Team did not observe any informative patterns from the samples , but is able to visualize how

a) encoder,decocder and cross attentions are working inside the t5 model &

b) feature engineered dimensions are affecting the attention layers