EE562A - Final Project

Students:

Elpida Karapepera, Max Ramstad

Project:

- Use the food-101 dataset to train a Vision Transformer model for classification.
    - Try at least two different classifiers.
- Utilize a pre-trained CLIP (ViT-32 base) in a zero-shot setting (without training anything) on the same dataset.
- Compare the performance of both models.
- Visualize the models' attention maps on a few images where CLIP does the correct classification and the trained Vision Transformer model fails and vice versa.

Implementation:

The implementation of the current project required:

- Training two different ViT classifiers on the dataset and evaluating their performance.
- Evaluating the performance of CLIP on the food-101 dataset.
- Comparing the accuracy of all the models with each other, which, to be done required:
    - Finding examples where each model fails, while the other models guess correctly.
    - Visualizing the attention maps of each model on the example images and commenting on the reasons why this happened.
    - Commenting on the overall efficiency of each model, after finding out its strengths and weaknesses.

Some background on CLIP and ViT:

In the CLIP model data (images and texts) enter two encoders (image and text respectively) extracting feature vectors, which are projected and normalized to form embeddings. These embeddings are scaled and compared with labels, to calculate each image’s cross-entropy cosine similarity towards each text (loss_i) and vice versa (loss_t). loss_i and loss_t averaged comprise the total loss, ensuring equality in training both modalities. The image encoder is a Vision Transformer itself.

ViT is designed purely for image classification tasks. ViT divides the image into patches and processes these patches solely through self-attention mechanisms. Each layer in the ViT model has multiple 'heads' in its multi-head self-attention mechanisms, with each head learning different aspects of attention over the patches.

Now what we are expecting to see:

As we mentioned, the image encoder of CLIP is actually a ViT model. This means that we are expecting the two models to perform very similarly when they are untrained. And this is exactly what happens! Taking a look at how the accuracy progresses in the ViT training we can see that the initial accuracy is 78%, while the CLIP accuracy is 77%, which is a pretty close match.

After training the ViT model specifically on the food-101 dataset, of course, we are getting much higher accuracy in the predictions.

Training ViT:

To train the ViT, we took a pre-trained vision transformer and finely tuned it on our dataset. The dataset had some noise and other variations already, so we didn't do much preprocessing to the images themselves. However to make the dataset compatible with our training module we needed to ensure that all of our images were RGB and not in greyscale. We did not find any overfitting when training it up to 15 epochs.

We trained:

- **'google/vit-base-patch16-224-in21k'**
  Where we got 84.8% in accuracy with 15 epochs.
- **'microsoft/swinv2-tiny-patch4-window8-256'**
  Where we got 80.2% in accuracy with 15 epochs and 84.5% in accuracy with 30 epochs (stoped training at 24 epochs due to overfitting).
- **'vit-large-patch16-224-in21k'**
  Where we got 20% in accuracy with 4 epochs and decided it was not worth training further, as all the other models started with at least 75% accuracy, without any training.

Differences between microsoft/swinv2-tiny-patch4-window8-256 and google/vit-base-patch16-224-in21k:
microsoft/swinv2-tiny-patch4-window8-256 uses a hierarchical transformer architecture with shifted windows, which allows for efficient computation and a smaller number of parameters compared to standard Vision Transformers like **google/vit-base-patch16-224-in21k**, which operates on fixed-sized patches of an image.

microsoft/swinv2-tiny-patch4-window8-256 divides the image into 4x4 patches and processes these in 8x8 windows. This smaller patch size might allow it to capture finer details, while **google/vit-base-patch16-224-in21k** divides the image into larger 16x16 patches. This makes it less sensitive to finer details but efficient for capturing more global information.

microsoft/swinv2-tiny-patch4-window8-256 is optimized for 256x256 input image size, while **google/vit-base-patch16-224-in21k** is optimized for 224x244 input image size.

In short:

microsoft/swinv2-tiny-patch4-window8-256 is generally better for tasks requiring understanding of local features due to its hierarchical nature and smaller patches, while **google/vit-base-patch16-224-in21k** excels in tasks where global context or larger receptive fields are beneficial. It's also heavily reliant on having large-scale datasets like the one that we are working with, for training to achieve its full potential.

microsoft/swinv2-tiny-patch4-window8-256 is designed to be more computationally efficient than standard **google/vit-base-patch16-224-in21k**, especially for higher resolution images.

CLIP evaluation in zero-shot setting:

To evaluate the CLIP model in zero-shot setting we loaded the model, found the image predictions for all the images in the validation set and then compared them to the correct labels, finding this way the percentage of right guesses, aka the accuracy of the model.
The model we used is "RN50" from https://github.com/openai/CLIP.git

Attention Maps Visualization - Comparing ViT and CLIP examples:

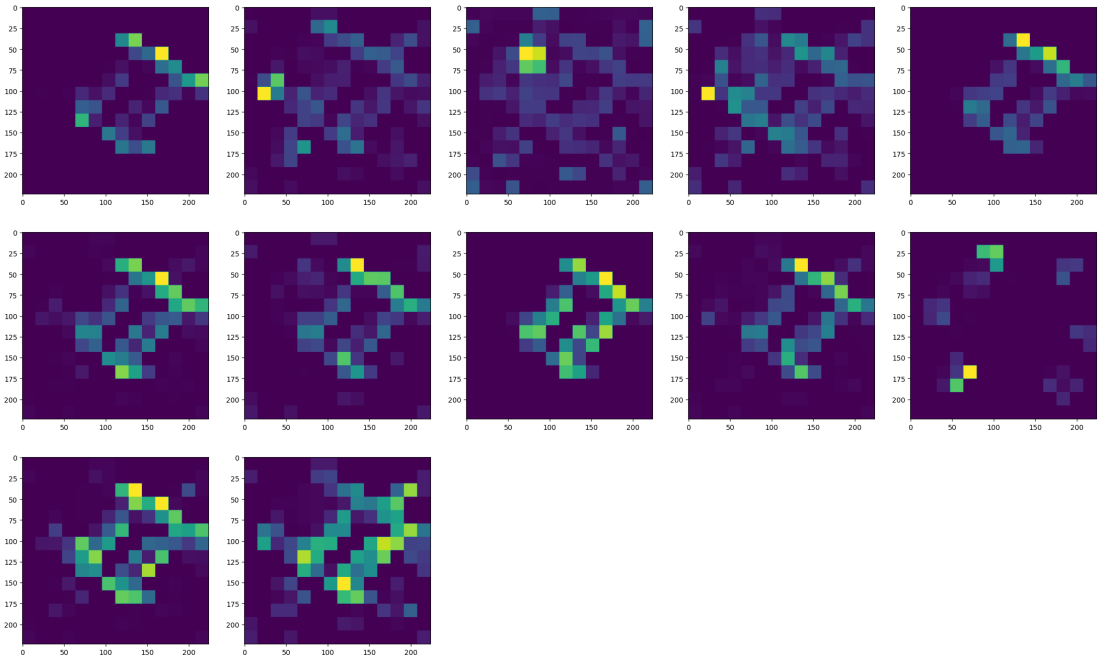
For the attention maps visualization we seeked for online resources and ended up finding some nice examples of visualizations in the google and facebook research websites ([1], [2]).

After loading both the CLIP pretrained model and the ViT model that we trained, we forward pass the validation dataset and get the predictions from both classifiers. We then compare the predictions with the correct labels and find all cases where CLIP is guessing wrong and ViT's guess is correct and vice versa. We then proceed with visualizing the attention maps of both models on the example images.

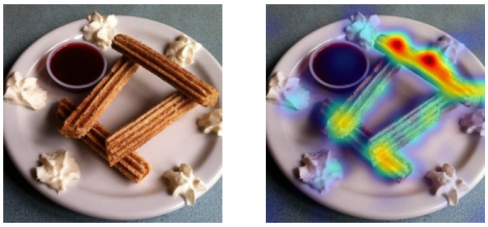
For visualization we had to finetune the visualization functions for each model separately, as the models have some fundamental differences between them. For example ViT simple ('google/vit-base-patch16-224-in21k') has a much larger attention map size (197) and thus requires a finer visualization grid (14x14) than the ViT swinv2 ('micorsoft/swinv2-tiny-patch4-windows8-256'), which is a smaller model with a smaller attention map size (63) and thus the visualization grid is smaller (8x8).

We capture the attention maps for the example images in both of the models and we visualize them.

The models have 12 attention maps each, which can be visualized separately (this visualization is commented out in the code). For example, here are the 12 attention maps of the simple ViT model that we trained in a picture:



We decided to depict them all together, in a single image (basically adding the heatmaps together):



In general, due to differences between the CLIP and the two different ViT models, readjusting the visualization maps to

- split the image correctly in windows and patches,
- pull the attention tensors,
- apply them as a gradient,
- scale them to fit the image size and
- combining them with the image to achieve the final result

took a lot of time and many readjustments.

Packages and Classifiers used:

All of our training was done in a Colab environment. We primarily relied on HuggingFace and PyTorch to train and finely tune our model. Other important packages were the openai CLIP model and matplotlib to plot our attention maps. For the CLIP attention map we edited and used https://github.com/hila-chefer/Transformer-MM-Explainability which edits the model and allows you to get the weights from the attention head. Similarly, we also used the Facebook Research group attention maps as well (https://github.com/facebookresearch).

As for the classifiers used, for the clip model we used the openai clip model (https://openai.com/research/clip). For the vision transformer we used the google vit-base-patch16-224-in21k and vit-large-patch16-224-in21k which is a pretrained model on the ImageNet-21k datase and the micorsoft/swinv2-tiny-patch4-windows8-256 model.

All packages: torch, torchvision, tensorboard, git+https://github.com/openai/CLIP.git, torchvision, pillow, matplotlib, transformers[torch], accelerat, datasets, transformers, evaluate, einops, torch, ftfy, timm, captum, scipy, scikit-image

Experiments:

Experiments were done in google colab pro+, in order to utilize online GPU power and perform long time training. We are provided with a rich dataset with a large amount of data, that requires a long time to process and get a model trained on it.

- Model Training: We trained the vit-base-patch16-224-in21k for 15 epochs and the micorsoft/swinv2-tiny-patch4-windows8-256 for 30 epochs using the cross-entropy loss function for both models.
- Model Evaluation: The models' performance was evaluated using as metrics the accuracy and the loss. We considered the evaluation metrics more important, as the training metrics can be due to overfitting (which is hopefully avoided as our model is using early stopping).
- Difficulties: Some images in the training dataset were black and white instead of RGB. This resulted in errors and interruptions during our training, as the trainer can only accept RGB images. We did not include those images in the training.

Results:

Training Loss and Accuracy:

- google/vit-base-patch16-224-in21k

Epoch	Training Loss	Validation Loss	Accuracy
1	1.131900	0.790138	0.784158
2	0.837700	0.706040	0.805822
3	0.556800	0.772385	0.795366
4	0.412100	0.811154	0.804832
5	0.404300	0.858802	0.805901
6	0.361300	0.943870	0.803446
7	0.120000	1.037089	0.807287
8	0.114700	1.045867	0.818218
9	0.068800	1.100100	0.821743
10	0.002600	1.150161	0.826851
11	0.005900	1.186047	0.828475
12	0.003400	1.189602	0.833188
13	0.011500	1.189824	0.841624
14	0.000000	1.167341	0.844911
15	0.000000	1.154160	0.847842

- micorsoft/swinv2-tiny-patch4-windows8-256


```
    },
    {
      "epoch": 30.0,
      "eval_accuracy": 0.870019801980198,
      "eval_loss": 1.3796367645263672,
      "eval_runtime": 406.0761,
      "eval_samples_per_second": 62.18,
      "eval_steps_per_second": 7.774,
      "step": 142050
    }
  ],
  "logging_steps": 10,
  "max_steps": 142050,
  "num_train_epochs": 30,
  "save_steps": 500,
  "total_flos": 7.413541354744971e+19,
  "trial_name": null,
  "trial_params": null
}
```


- vit-large-patch16-224-in21k

```
***** eval metrics *****
  epoch                   =        4.0
  eval_accuracy           =     0.2501
  eval_loss               =     3.1002
  eval_runtime            = 0:19:40.12
  eval_samples_per_second =     21.396
  eval_steps_per_second   =      2.675
```

Evaluation Metrics:

- google/vit-base-patch16-224-in21k

```
***** eval metrics *****
  epoch                   =       15.0
  eval_accuracy           =     0.8058
  eval_loss               =      0.706
  eval_runtime            = 0:07:49.31
  eval_samples_per_second =     53.802
  eval_steps_per_second   =      6.727
```

- micorsoft/swinv2-tiny-patch4-windows8-256


```
    },
    {
      "epoch": 30.0,
      "eval_accuracy": 0.870019801980198,
      "eval_loss": 1.3796367645263672,
      "eval_runtime": 406.0761,
      "eval_samples_per_second": 62.18,
      "eval_steps_per_second": 7.774,
      "step": 142050
    }
  ],
  "logging_steps": 10,
  "max_steps": 142050,
  "num_train_epochs": 30,
  "save_steps": 500,
  "total_flos": 7.413541354744971e+19,
  "trial_name": null,
  "trial_params": null
}
```


- vit-large-patch16-224-in21k

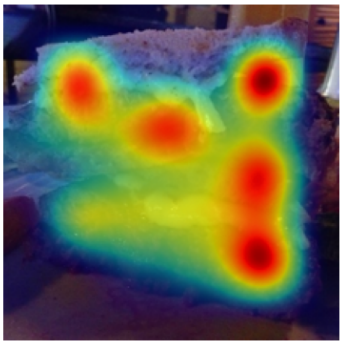
```
***** eval metrics *****
  epoch                   =        4.0
  eval_accuracy           =     0.2501
  eval_loss               =     3.1002
  eval_runtime            = 0:19:40.12
  eval_samples_per_second =     21.396
  eval_steps_per_second   =      2.675
```

Attention Maps:

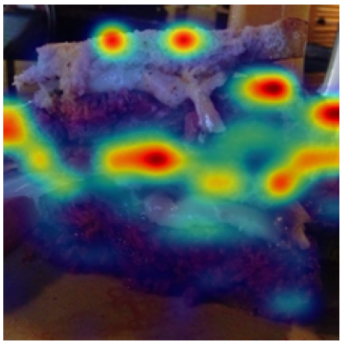
Clip guesses correctly 540 images that ViTs missed
ViT guesses correctly 593 images that CLIP and ViT swinv2 missed
ViT swinv2 guesses correctly 1742 images that CLIP and ViT missed

Plotting 3 examples where CLIP is performing better than both ViT models:

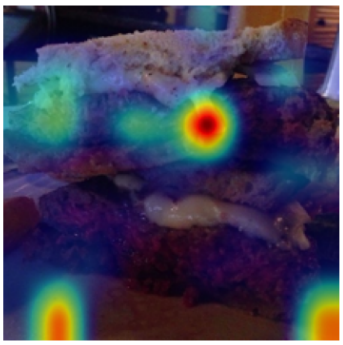
Image 1:
True Label: hamburger
CLIP Prediction: hamburger

ViT Prediction: pulled_pork_sandwich

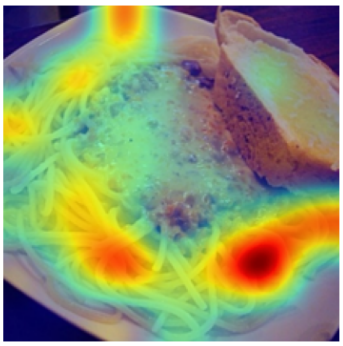



ViT swinv2 Prediction: beef_tartare

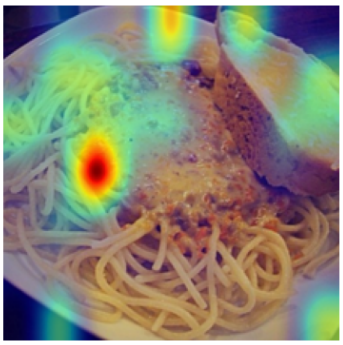
We can see that both pulled_pork_sandwich and beef_tartare are guessings similar to the correct answer (hamburger).
The google/vit-base-patch16-224-in21k captured both the meat and the bun, guessing pulled_pork_sandwich and the micorsoft/swinv2-tiny-patch4-windows8-256 focused on the meat, guessing beef_tartare. As mentioned before, micorsoft/swinv2-tiny-patch4-windows8-256 divides the image into 8x8 overlapping windows and thus focuses on smaller details on the image, rather than the bigger overall picture. Thus, it makes sense to capture only parts of the hamburger (the meat inside it) and make a guess based on those observations.

Image 2:
True Label: spaghetti_carbonara
CLIP Prediction: spaghetti_carbonara

ViT Prediction: spaghetti_bolognese

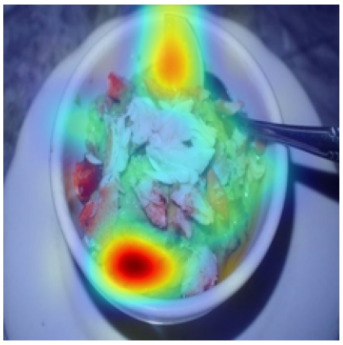

ViT swinv2 Prediction: spaghetti_bolognese

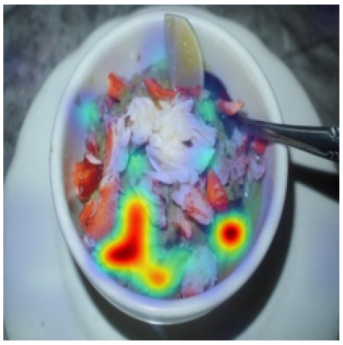
We can see that here clearly how the CLIP model focuses on the big picture, which is the pasta, first and secondary observations are the toppings.
The google/vit-base-patch16-224-in21k and the micorsoft/swinv2-tiny-patch4-windows8-256 focuses on smaller details instead.

Plotting 3 examples where ViT simple is performing better than both CLIP and ViT swinv2 model:

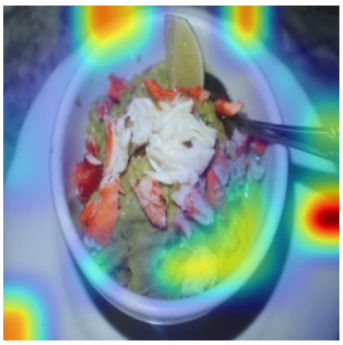
Image 1:
True Label: guacamole
CLIP Prediction: ceviche

ViT Prediction: guacamole

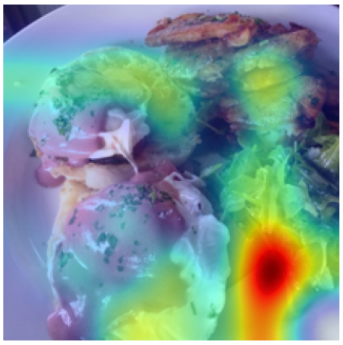



ViT swinv2 Prediction: greek_salad

The CLIP model observes everything inside the bowl and, since a lot of ingredients can also be found in a greek salad, guesses greek_salad.
The google/vit-base-patch16-224-in21k spots the guacamole among the ingredients and makes the right guess.
The micorsoft/swinv2-tiny-patch4-windows8-256 shares its attention among the guacamole, and the plates, providing an incorrect guess.

Image 2:
True Label: eggs_benedict
CLIP Prediction: crab_cakes

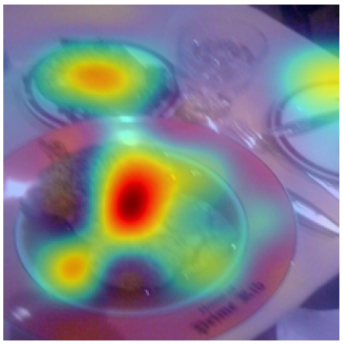
ViT Prediction: eggs_benedict


ViT swinv2 Prediction: shrimp_and_grits

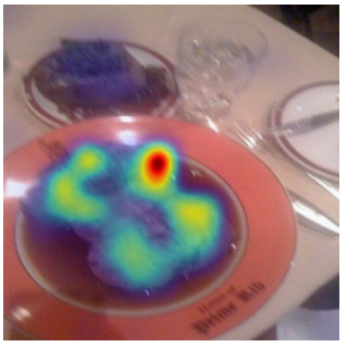

The CLIP model observes everything including the side salad and the bread, not paying much attention to the eggs themselves.
The google/vit-base-patch16-224-in21k and the micorsoft/swinv2-tiny-patch4-windows8-256 focus again on smaller details in the picture as expected.

Plotting 3 examples where ViT swinv2 is performing better than both CLIP and ViT simple model:

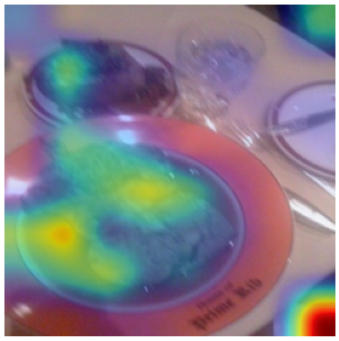
Image 1:
True Label: prime_rib
CLIP Prediction: filet_mignon

ViT Prediction: sashimi

ViT swinv2 Prediction: prime_rib

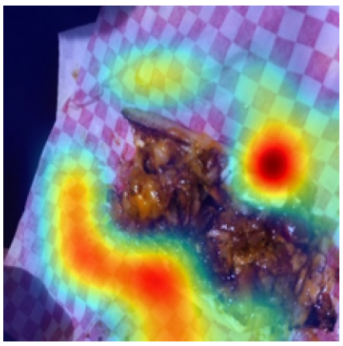



All models here are pointing their attentions towards the right part of the image, which is the meat.

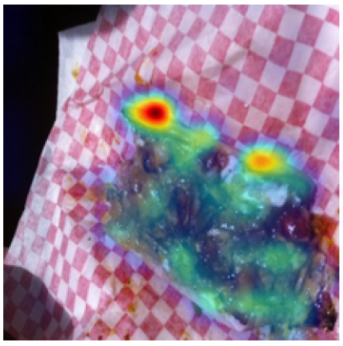
Image 2:

True Label: hot_dog

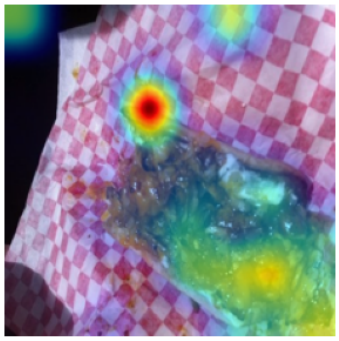
CLIP Prediction: pulled_pork_sandwich

ViT Prediction: takoyaki

ViT swinv2 Prediction: hot_dog

The CLIP model herecaptures the bread of the hot dog and provides a close sandwich prediction, pulled_pork_sandwich.
The google/vit-base-patch16-224-in21k captures a lot of the toppings of the hotdog, that could look like the toppings usually seen on takoyaki.
micorsoft/swinv2-tiny-patch4-windows8-256 successfully captures both the bun and the sausage of the hot dog.

Conclusions:

As we somewhat expected. The finely tuned ViT models were more accurate than the zero-shot CLIP model. The biggest downside of the finely tuned ViT was the training time. We attempted to do the “large” ViT model but 4 epochs took ~10 hours and was nowhere near the accuracy we would be looking for.

The trained google/vit-base-patch16-224-in21k seemed to perform the best out of all models.

Contribution:

Elpida:

- CLIP implementation
- ViT swinv2 implementation (microsoft/swinv2-tiny-patch4-window8-256)
- Comparison of the ViT simple vs ViT swinv2 vs CLIP to obtain the image examples
- Attention maps visualization implementation for:
    - CLIP (version #1)
    - ViT simple: google/vit-base-patch16-224-in21k
    - ViT swinv2: microsoft/swinv2-tiny-patch4-window8-256

Max:

- ViT simple implementation (google/vit-base-patch16-224-in21k)
- Found and fixed dataset bug (greyscale images) - dataset modification
- CLIP attention map visualization (version #2)

Both:

- Report
- Presentation

Code Google Collab files:

- **Training ViT code (including the links to the collabs we used to train the other ViT models, the CLIP model and perform the visualization of attention maps on images):** https://colab.research.google.com/drive/1NDA4f6dxc_I2vboEdjBWVG5WOw9nBWP0?usp=sharing
- **Training the ViT simple and large:** https://colab.research.google.com/drive/1mu1z-SDgGbJFD1M7jLxVrAzsm6xu3Wta?usp=sharing
- **Training swinv2 ViT:** https://colab.research.google.com/drive/1Hp7coeC0eGtW1WgfaPoEZuwhpR7rJ7j2?usp=sharing
- **Visualization of attention maps:** https://colab.research.google.com/drive/1N4wOUsFzKzG_XUuMpszpDFa6mt7ZzO2W?usp=sharing

References:

- **Packages and Classifiers:** torch, torchvision, tensorboard, git+https://github.com/openai/CLIP.git, torchvision, pillow, matplotlib, transformers[torch], accelerat, datasets, transformers, evaluate, einops, torch, ftfy, timm, captum, scipy, scikit-image
    - https://github.com/hila-chefer/Transformer-MM-Explainability
- **[1] Attention Map Visualizations separate:** https://github.com/facebookresearch/dino/blob/main/visualize_attention.py
- **[2] Attention Map Visualizations combined:** https://colab.research.google.com/github/kevinzakka/clip_playground/blob/main/CLIP_GradCAM_Visualization.ipynb