Project Final Phase

Eliada Polydorou
Christoforos Seas

Abstract

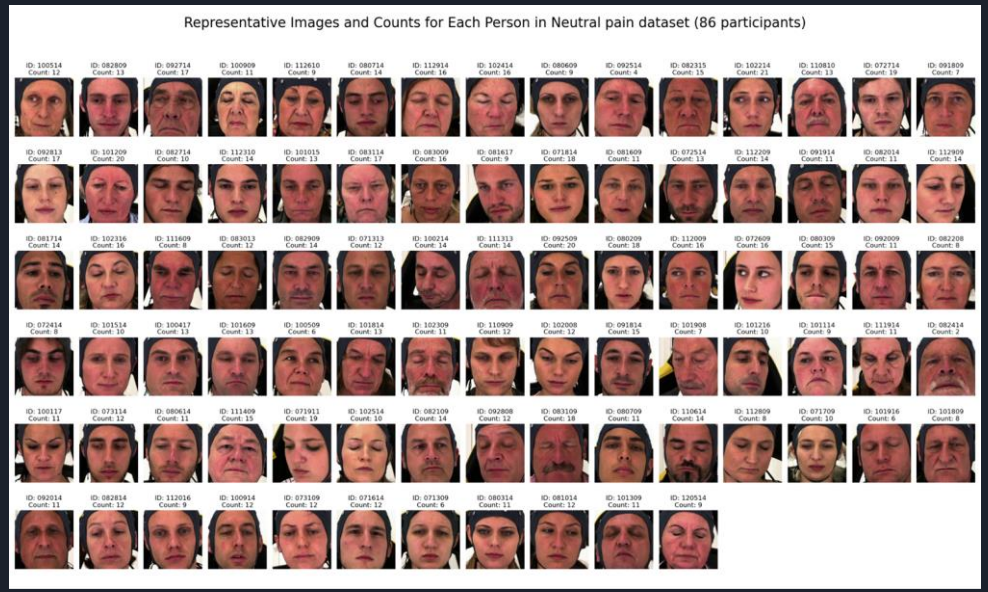
This study explores using vision transformers to distinguish pain and no-pain images in the BioVid dataset. Evaluating on a subset of 1000 images per class showed promising results for pain detection. However, the findings warrant caution due to dataset limitations. This emphasizes the crucial need for well-curated datasets in ensuring reliable machine learning models, particularly in sensitive domains like detecting pain from visual cues.

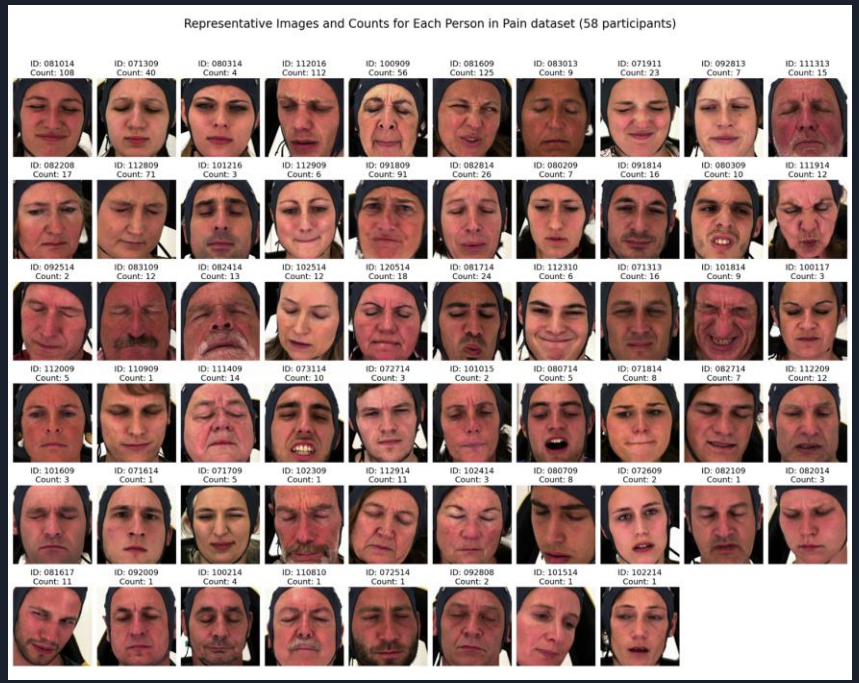
Our process

Our task was to use the vanilla Vision Transformer (ViT) architecture, in our BioVid dataset to create a model that can distinguish pain vs no-pain images. However, as we mentioned in the previous presentation (link), this proved to be too time-consuming and wasn't promising enough, so we shifted our focus to more efficient variants like TinyViT, CNN and FastViT due to their superior speeds and performances.

First experiment

Initially, we took a look at our dataset to see what's going on, and whether it would be good to do any data curation. We visualized the dataset as shown in the pictures below, to have a clear image on what we are working on.

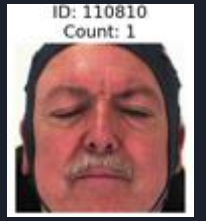


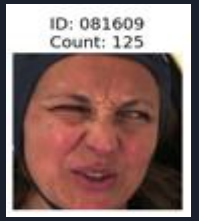


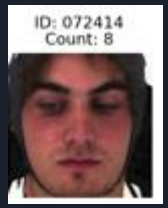
Comments on the dataset

From the dataset, we can observe the following details:

1. The total amount of images is 1000 for pain and 1060 for no-pain
2. There are more participants in no-pain vs in pain (86 vs 58)
3. The amount of images of each participant varies from 1 (e.g., 110810, pain) to 125 (081609, pain)
4. The no-pain dataset is more equally distributed across the participants. There are 19 participants with 3 or less frames in pain dataset (high standard deviation), and 6 with 40 or more, whereas the frames for each participant in no-pain dataset is approximately 8 to 20 (low standard deviation)
5. Some participants appear only in one dataset (e.g., 072414 in no-pain)
6. Many of the participants have more images in one dataset (e.g., 112016 has 112 images in pain and 9 images in no-pain)
7. Some images are hard to be distinguished. A human would possibly categorize them wrong (e.g., 083109, 112914)







Pain

No-pain

Possible training problems based on the dataset

1. Since some participants appear only in one dataset, if the model learns the facial characteristics of these specific participants, it will be 100% accurate. In other words, the model will possibly predict “pain” or “no-pain”, just by seeing the participant’s face. For instance, it may learn to always predict “no-pain” for participant 072414, because they appear only in no-pain dataset, and it will be correct 100% of the times. Also, for participants that have different amount of images in each dataset, it may predict (again) one class, based only on facial characteristics. For instance, if it always predicts “pain” for 112016, it will be correct 92.6% of the times, since it has 112 images in pain and 9 images in no-pain.
2. The pain dataset has high standard deviation, so the model is trained mainly on a few faces, the ones who have greater occurrence, meaning that it’s more likely to learn characteristics of faces rather than characteristics of pain, since it doesn’t have much face variance
3. It’s difficult even for a human to classify the images. Given also that they are just 1000 for each class, our model will probably struggle to understand the differences between pain and no-pain

What we did

We initially tried two experiments. For each experiment, we split our dataset into 80% training data, 10% validation data and 10% testing data. We used TinyViT, CNN, and FastViT and ran it for 10 epochs. The results from the three different methods were very close. We used 1000 no-pain images instead of 1060.

Subjective Split:

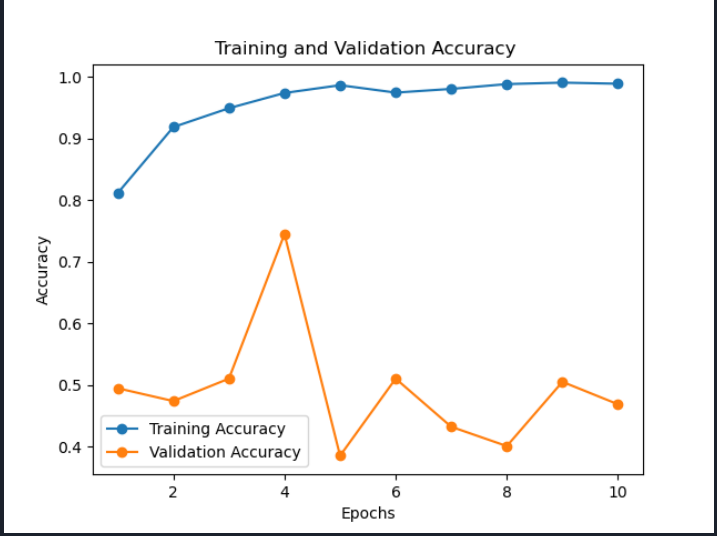
We divided our dataset so each participant DOES NOT appear in another dataset. We used this methodology, because we wanted our model to be subjective and not learn the faces, but learn the pain and no-pain instead.

Participants Split:

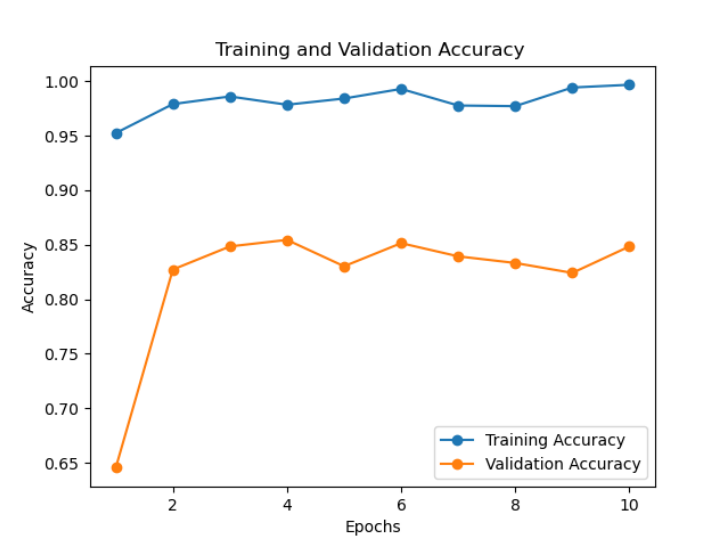
We divided our dataset so each participant appear both in training and validation data, and if possible in testing data (if the participant had 10 or more frames). We used this methodology, because we wanted a) to see whether the model learns the faces instead of emotions, and b) to be more certain that our model is training properly.

Results (From FastViT)

Subjective split



Participants split



We can observe that for both the subjective and participants split, the training accuracy is very high. However, the validation accuracy is very low in subjective split, but very high in participants split. The testing datasets accuracies are:

Subjective Split: 51%

Participants split: 94% (!)

Further experimentation

Since our model didn't yield good results, we tried our best to manage to create something useful. After trying different optimizers, activation functions, etc., we proceeded to try different dataset split. More precisely, we splitted the whole dataset into 80% training and 20% validation 5 times. Each time the participants were different, so no participants would appear both on training and validation dataset. Moreover, each dataset had different participants, and different combination of amounts of participants. For instance, one dataset has only two participants that contain many frames each, other has many frames of one participant and a few of others, and another dataset has many participants with few frames each.

Model details

We used only FastViT to train our models, because the other methods yielded very close results, and FastViT proved to be indeed, as the name suggests, very fast.

The model training setup has an input shape of (256,256,3). We replaced the original output layer with a new Dense layer with 2 neurons using sigmoid activation function for generating probability scores for the pain and no-pain classes. The model was complied with ‘adam’ optimizer and ‘categorical_crossentropy’ as the loss function. The batch size of 32 was employed during training for computational efficiency.

While training our models, we realized that the `pretrained=True` parameter played a major role. When we used the pretrained model, we essentially used the model with weights adjusted for the ImageNet (https://www.image-net.org/) classes.

Therefore, we decided to trained our models, both using `pretrained=True` and `pretrained=False`.

Results

Our model managed to yield good results, with our highest validation accuracy reaching 88%!

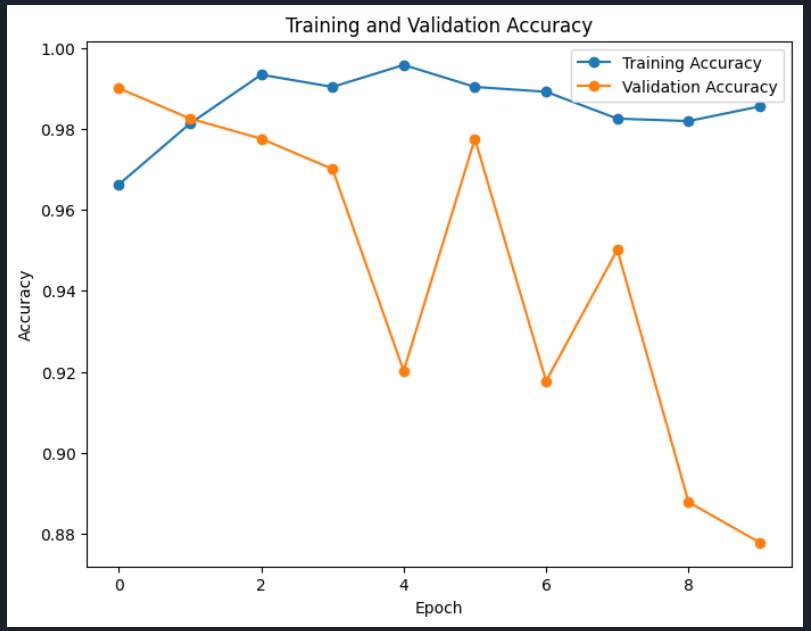
The difference between running with adjusted weights or not, vary from 1% to 36%. The table below shows the accuracy on the validation set, after the model finished training.

TABLE I: Validation accuracy on different datasets

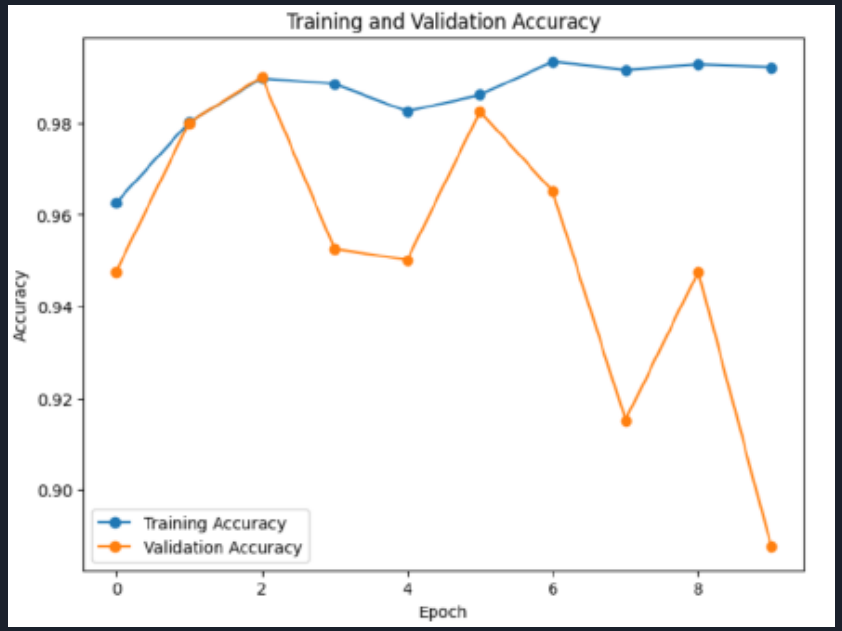
Dataset number	Pretrained True	Pretrained False
1	0.66	0.69
2	0.79	0.81
3	0.81	0.85
4	0.87	0.88
5	0.87	0.51
Average	0.80	0.75

More on accuracy results

`pretrained=False`



`pretrained=True`



Metrics

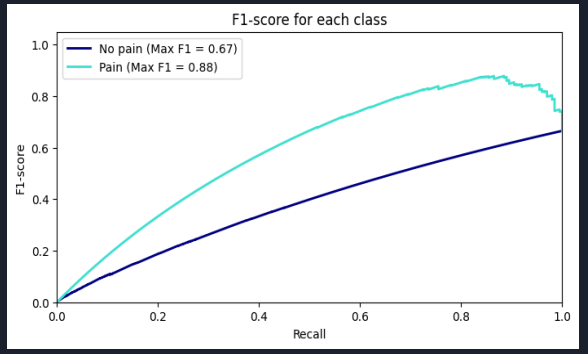
Our accuracy results may indicate a great performance. However, it’s important to see other metrics, such as F1 score, precision-recall graphs and ROC graphs.

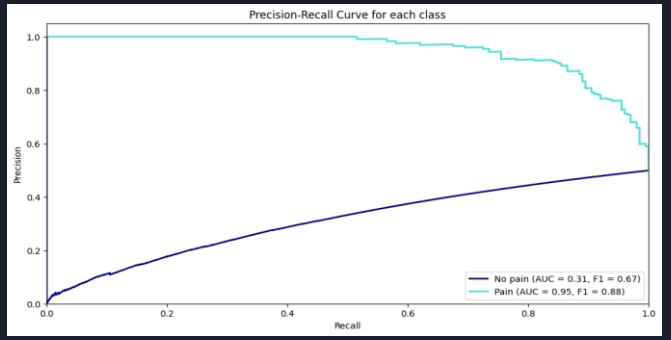
Our best run was number 4 with `pretrained=False`.

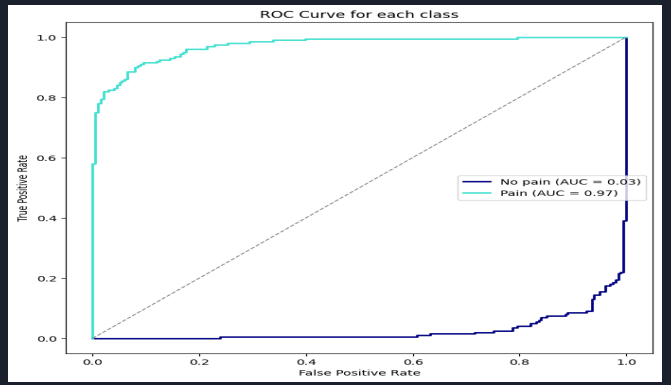
Graphs:

- The model achieved an F1-score of 0.88 for pain and 0.69 for no-pain. This suggests that the model is able to identify pain in facial expressions with relatively high accuracy, but it is struggling more to identify no-pain. We can also see this with the other graphs as well, since pain has way better results than no-pain.

- The model achieved high precision for pain expressions at all levels of recall. That means that the model is very good at identifying pain expressions, even when the recall is high(meaning it can identify a large portion of actual pain expressions). It seems that the neural network is struggling to identify no pain expressions at low levels of recall. We can observe the same with the ROC curve, where the no-pain class has very low true positive rate when the false positive rate is low as well.



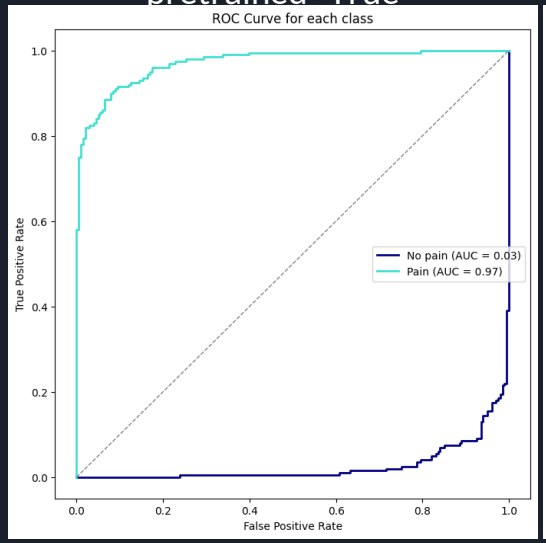




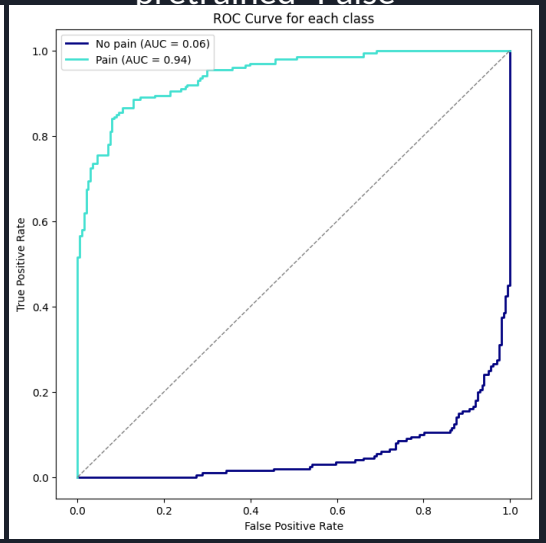
Let’s take a closer look

Let’s compare the graphs with `pretrained=False` and `pretrained=True`

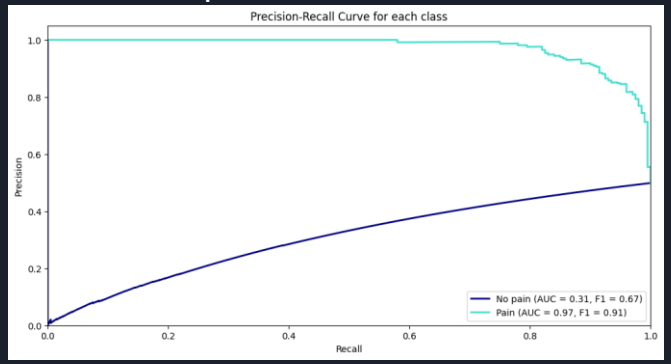
pretrained=True



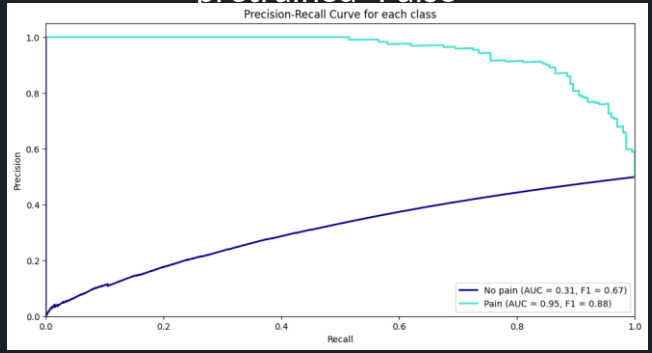
pretrained=False



pretrained=True



pretrained=False



Both models are more pain-biased. However, the pretrained model is even more biased. We can conclude that our models are good at identifying pain, but not good enough to identify no-pain. Possible reasons for that are the dataset’s possible problems as mentioned above.

Discussions

This model may be proven vital, because maybe it's more crucial especially for health experts to identify pain.

Another possibility is that the neural network is not able to capture all of the subtle nuances of facial expressions that can be used to distinguish between pain and no pain. For example, the neural network may not be able to distinguish between a slight frown and a frown that indicates pain.

Possible case scenario: Someone lies on the hospital bed, and there's a camera which sends frames to our model to predict. Our model would probably identify pain if the patient feels pain indeed, but it's also highly possible that it would predict that the patient is in pain, even if he is not. When the model identifies pain, it can notify the doctor or nurse to see the camera and check. Better safe than sorry, so if the model mistook a picture as pain it's not very important.

What we have learned

Possibly the most important part for the implementation of a machine learning model that can classify what we want is a good dataset. Furthermore, doing a data analysis is also important. As we saw, the same dataset with different split methods can change the accuracy results from 50% to 94%. A wrong split can mislead us on believing we managed to create a great model, but in practice that model would be proven inefficient.

It’s also crucial to take into account many metrics such as F1 score, not just than accuracy results, to comprehend the results better and see the correlation between the model’s predictions.

Thank you