Machine Vision: Assignment II
(Lab Practical 10)

10th December, 2025

1 Introduction

After our 10th weekly in-person lab, you should complete the task and submit your work as the second (of two) marked courseworks for Machine Vision. This document sets out the directions that apply for both the lab and the assignment. **Task:** We want you to design, train, and submit a model which takes as input a video (in .mp4 format), and outputs an integer: count the number of pushups which were *completed* in the video.[1]

Despite huge advances (for example Veo-3, Sora, Flux 2, and many more), state-of-the-art AI models still struggle significantly at video-understanding. Particularly when asked questions about observed dynamics.

We do not expect you to 'solve' this problem. In fact, we will be awarding relatively few marks for absolute model-performance. We are *much* more interested in *how you approach* this problem. See the Project Report section for more details on how to submit your work, and the Grading section for specific information on how your work will be marked in this assignment.

1.1 AI-tools

In your previous assignment, it was expressly prohibited for you to use AI tools to complete the assignment.

In this assignment, we now actively encourage you to use AI tools, **with a caveat**. As long as you don't discuss the task itself on Moodle, it is OK (even encouraged) to share and discuss information about code LLM's and coding environments on the shared Moodle forum. E.g. If the coding LLM built into Colab (Gemini) isn't available, there are other options besides cut-and-pasting, and some are free, though with restrictions.

However, now that we're permitting you to use AI models to help you complete an assignment (something which previous cohorts that took this course did not have access to), we naturally

[1] To be a bit more explicit here: we are defining a pushup as the motion which starts from full extension of one's arms, and ends with a subsequent full extension of one's arms.

expect a far higher standard of output.

1.1.1 The Caveat

You are free to brainstorm your approach to this assignment with an LLM. You are also free to use an LLM to help you write code to complete this assignment. However, you **must not** use an LLM to write your project report.

It's a very good thing for you to use state of the art (SOTA) AI models to push the frontiers of your capabilities in designing and training models. However, that cannot come at the expense of your fundamental understanding of what you are doing. We are looking to see creative, nuanced, and interesting write-ups in the Project Reports. We want to hear about *your* efforts and iterations in *your own words*. We do *not* want to read AI-generated slop reports. If we detect that you have used an LLM to write your Project Report, you will score 0% on this assignment, and be subject to the UCL disciplinary procedure on cheating.

2 Deliverable

One of the goals of this assignment, is to get you comfortable with using some of the tools and techniques that are useful for a career in computer vision (be that in industry, or in academia).

The thing you *actually* need to *submit* on Moodle for the January deadline is exactly one PDF Project Report. However, that Project Report must clearly contain **public** links to your code that you used to complete this assignment. Only share the public link with the Instructor/TA's by linking to it in the report - do NOT share it with others obviously! Do not modify the code any further once submitted, because the latest time-stamp (among the report, model, and each file of code) will be used as your time-of-submission for lateness purposes. Specific instructions for how to submit your code are below.

2.1 Code

You must submit all of the code you used to complete this assignment.

All code must be submitted inside **your** copies of these two Google Colab notebooks:

1. Model Training Code
2. Model Inference Code

Please open each notebook in turn, go to 'File', 'Save a copy in Drive' (or, alternatively, 'Save a copy in GitHub'), and proceed to fill in your code in-line. For saving in Google Drive, Google sign-in will be required. Optional, and we have not tested this yet, but you may be interested in the Google Colab Extension that lets you work in VS Code but run on Google's computers.

When it comes to time of submission, make your Colab notebooks public, and clearly include the links to your notebooks in your Project Report. Failure to clearly include the links to your completed versions of the above notebooks will result in you scoring zero marks for the Model Performance section.

2.1.1 Model Training Code

This notebook should contain all of the code necessary to train your model from scratch, save it to disk, and, upload the model to a public repo on HuggingFace.

Starter code is provided for you in this notebook, and a number of TODOs are set for you throughout. Completion of all of the given TODOs is the minimum requirement. The provided TODOs show a rough skeleton for a basic implementation of this assignment. The expectation is that students will invent their own model or combination of models to address this task. Think in terms of 3D, but also 2D or 1D.

If you find you need to add more cells to this notebook to carry out the analysis necessary to write your Project Report, that is completely welcome. Please feel free to add as many cells as you see fit. Please note, however, we expect to see all plots, analysis, and discussion for this assignment in your Project Report. Please write the **code** you need in the notebook, but please keep all of your substantive analysis in one place: the Project Report. We do not want to be flicking back and forth endlessly between your code and your report. We will spend the vast majority of our time reading your report, and we will only go through your code afterwards, mainly just to make sure you have actually done the work.

Your notebook, as submitted, should 'just work'. That is to say, a TA should be able to open the notebook, click 'Run all', and observe your model train. Double, triple, even quadruple check that this is the case. If this is *not* the case, you will score 0 marks in the Model Performance section.

2.1.2 Model Inference Code

This notebook is much more brief, and there is a lot less for you to do.

This notebook will download your model from HuggingFace, and evaluate it against a set of video data. The TAs will use this exact notebook to grade your model performance.

You do not have access to the test data, naturally, but it is your responsibility to make the notebook run without errors when evaluating against the provided training data. As long as you ensure this is the case, you can be safe in the knowledge your code will work when the TAs change the target to be the test dataset.

While there are very few TODOs for you to complete in this notebook, it is paramount you take due care and attention to make sure that this notebook will work with your chosen model architecture. Failure to do so will result in you scoring zero marks on the Model Performance section.

2.2 Model

2.2.1 Model Weights

You are **not** permitted to consume any model via API request. For example, you are not permitted to submit a video to Gemini 3.0, and prompt it to count the number of pushups completed in the video[2].

You **are** permitted to use pre-trained model weights. So, for example, you *may* try and construct a model on top of Dinov2, or Dinov3, or SAM3D Body, provided that the combined footprint of your model does not exceed the memory footprint requirement.

Put another way, all weights of *any* model you use must be loaded by your code onto your device, and the combined footprint of the model(s) on your device must not exceed the corresponding model footprint.

2.2.2 Model Submission

Your model must be made public via HuggingFace Hub. You should make a HuggingFace account (if you don't have one already) and upload your model to a public repository. This step must be clearly completed in the Model Training notebook. Code is provided to help you here.

You should also clearly reference a **public** link to your model in your Project Report.

2.2.3 Footprint

Your model, at *inference* time, must not have a memory footprint (RAM) of more than 12GB.

Your model, at *training* time, must not have a memory footprint (RAM) of more than 20GB.

2.2.4 Training

The model training code you submit in your Model Training Colab notebook, must run, from start to finish, on the training data which you have been provided, in under 5 hours, on an Nvidia L4 GPU.

That is to say, a TA should be able to open your Model Training Colab notebook, select an L4 runtime, click 'Run all', come back in 5 hours, and see that execution has ended, and your model has fully trained.

This **includes** any time taken for data preprocessing necessary for your model to train. As an explicit example: say you were to use a pose-estimation model to extract trajectories of pose data from the videos in the training data, and that takes 3 hours to complete, and then you were to subsequently train a model to estimate the number of completed pushups from the pose trajectories, and that took 3 hours, then that would count as 6 hours total execution time, and that would be over the time constraint.

[2] Gemini 3.0 actually performs very poorly at this task, so you probably wouldn't want to try this anyway.

You can also (but are not required to) include some code in your training loop which prints a UNIX timestamp at:

- The beginning of training.
- A semi-regular cadence throughout training.
- The end of training.

This can also help convince the TAs that your model trains within the required time constraint.

If your model does not train from start-to-finish within the time constraint, you will score zero marks on the Model Performance section.

2.2.5 Latency

Your model must be able to operate over 100 frames of test video in under 60 seconds, when run on an Nvidia L4 GPU.

If it does not, you will score zero marks on the Model Performance section.

2.3 Training Data

The training data can be accessed by running the first cell in the provided Model Training notebook. It is hosted in a public S3 bucket.

You are not permitted to add additional *new* videos to the training data, however, you *are* permitted to *augment* the provided videos.

We are particularly interested in discussion and evaluation regarding your approach here. For more information see 2.5.1.

2.4 Test Data

The test data is, understandably, not provided to you. The TAs will use the held-out test data to test your model performance.

You are told the following key pieces of information about the test data, however:

Every video in the test data will comply with the below constraints:

- Under 1,000 frames in length.
- Under 1080p resolution.
- Only contain one person.
- Only contain whole, distinct (integer units of) pushups.

Furthermore, you are made aware that there will not be any videos in the test data which contain more than 10 pushups. Should that persuade you to reframe the problem as a classification, rather than a regression problem, that is your prerogative.

2.5 Project Report

This is the most important part of your submission, and is where we will be awarding the majority of the marks for this assignment. The format for this report is quite unique. We are looking for a very specific type of report. Please pay close attention to the instructions given in this section.

The clue is in the name: Project **Report**. We are looking for a linear, descriptive, journal of your work on this project. We *want* to hear all of the details. We are *particularly* interested in content like:

1. I tried x because I thought y.
2. I evaluated my implementation of x by doing z.
3. Here are some plots of my evaluation metrics, and some discussion around why I chose them .
4. Surprisingly, it turned out, that actually my belief in y was misguided. In actual fact z is true (insert some novel realisation you happened upon by virtue of inspecting your results).
5. Therefore, I rearchitected my model in light of z, and, saw greatly improved performance, as evidenced by these informative plots of yet more evaluation metrics.

Note, we are *not* looking for something resembling a paper here. A paper is designed to place the majority of the emphasis on the final product: the ground-breaking innovation, dataset, or architecture that the authors have come up with. In contrast, in this assignment, we care *very little* about your final product. We care *far more* about your experimental process. We *want* to see interesting evaluation metrics, thoughtful discussion about your architecture, and creative approaches around augmenting the training data. While marks will be awarded for model performance, comparatively, you can score far more marks by producing a well-written project report, than a well-performing model.

A good report, would probably cover the below topics. This is not a prescriptive structure (feel free to structure the report however you like).

2.5.1 Data Augmentation

As stated above, you are **not** permitted to *add extra videos to the training data.* However, you **are** permitted (and are encouraged) to **augment** the training data. The graphic in Figure 1 from this paper is quite informative in this respect.

Put another way, you can perturb, translate, rotate, scale, mask, or make any modification you like to any of the provided videos, but you cannot add extra videos to the training data.

2.5.2 Data Preparation

Before feeding your video data into your model, you will need to make some decisions as to how you transform your .mp4 data into data your model can understand. We are very

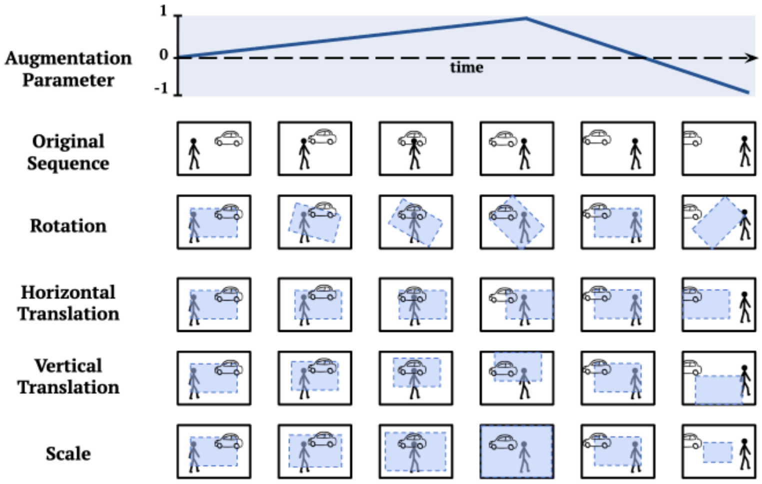

Figure 1: Data Augmentation Techniques

interested in your reasoning around your approach here.

As an idea of some things to think about (you don't have to address all of these points, these are just suggestions):

1. **Frame rate**. Are all of the videos in the training data the same frame rate? If you're using any pretrained models, what frame rates do they operate over? What is the effect of varying frame rate on the final accuracy of your model?
2. **Resolution**. Are all of the videos in the training data the same resolution? If you're using any pretrained models, what resolution do they operate over? What is the effect of varying resolution on the final accuracy of your model?
3. **Positional Encoding**. Dependent on your choice of model architecture, you might want to carry out some analysis on the impact of positional encodings (and choice thereof) on your model performance.
4. **Video Length**. How does your model cope with videos of varying lengths? If the input dimension of your model is fixed, what do you do? Do you then use padding in your input? Is that not a lot of wasted computation if you end up attending over padding tokens? Would you implement an attention mask in that case?

2.5.3 Model Architecture

Starting with a concise summary of your architecture is generally a good idea.

For example: 'my final model uses a pre-trained 3D pose estimation model to detect trajectories of body pose, before passing the resultant sequence of pose data through a transformer architecture, which I trained from scratch, to output an integer count of pushups'.

Then, please walk us through each component of your model in detail. Please pay particular attention to how the different components of your model connect to each other.

Please be sure to walk us through the logic for your approach. We would love to see examples of ‘failed’ approaches. N.B. don’t just tell us you tried something and it didn’t work. Show us. We are looking for well reasoned arguments as to *why you thought some approach was going to work*, and then, some interesting plots of evaluation metrics which led to your initial belief being confounded, followed by a discussion of how you updated your beliefs / approach in light of the evidence presented. For example: ‘I first started out with a 2D pose estimation model, but the estimated pose trajectories were not accurate enough, especially when dealing with partially occluded video (see my 2D vs 3D pose-estimation evaluation section for more analysis on this). I switched to a 3D model and found that the improvement in performance justified the increase in model footprint, and in latency (see my latency evaluation section for more analysis on this)’.

It would also be an *extremely* good idea for you to include a diagram of your model architecture. Multiple examples abound in the literature (it is a rarity to see a paper without a model diagram). It’s usually also useful to see some explanation of how gradients flow through your model, in your diagram.

2.5.4 Evaluation

This part of your Project Report should contain some overall evaluation of your final model.

Note, this differs slightly from the evaluation you will have shown up until this point. Whereas beforehand, we were looking for you to show some evaluation which would have guided you in making a decision between various model architectures, or data pre-processing techniques, in this section, we would like to see a summative evaluation of your final model performance and characteristics.

Things to include here would be:

1. Model footprint.
2. Model latency.
3. Edge cases / failure cases your model does not perform well on.
4. Some visualisation of the activations of your intermediate model layers on different inputs.

2.5.5 Extensions / Future Work

If you have some insight as to potentially interesting extensions to your model, which you would like to attempt, if you had (A) more time, (B) more data, (C) more compute, (D) all of the above... Then please write about it here.

3 Grading

This assignment will be graded out of 100 marks. NOTE: You must include two things from your code in the report, to show that your report lines up with your code and your experiments (the SUMMARY and your random seed) - see below. The breakdown of how marks are awarded will be as follows:

3.1 Model Performance

20 marks.

Here's how they'll be awarded:

We will open your Model Inference Code, and will select an Nvidia L4 runtime. We will click "Run all". If your code doesn't make it past this point (if it errors out already), you will score 0 out of 15 marks.

We will then change the input source to be our own held out test set. This will be a folder of .mp4 files, compressed with the H.264 codec. The test data will respect the constraints outlined in section 2.4. The only thing we will change is a bucket name, and your code should 'just work'. If your code runs correctly, you will score 5 marks out of 20 (regardless of your latency or accuracy figures).

To get the most out of the 15 marks, 5 marks will be awarded for latency, and 10 marks will be awarded for accuracy on the test-set. Specifics: we will not be providing target latency / accuracy figures. As a rough guide however, we will count each prediction as 'correct' if and only if it matches the target number exactly. Unless your model scores over 55% accuracy averaged over the test-set, you are most likely scoring 0 out of the 10 marks for accuracy.

There are no explicit marks for *this* section being *awarded* for your Model Training code. That is by design. However, you can *lose* marks for this section due to your Model Training code (if it takes too long to train, or doesn't train at all). This section is for rewarding the *performance* of your model, provided that your model met the specified training constraints.[3]

If your code exceeds the strict model-footprint limit, and / or exceeds the latency limit, you will score 0 marks on this section.

3.2 Clean Code

10 marks.

If your submitted code is easy to follow, not excessively verbose, and does *not* look like you just copy-pasted it from ChatGPT, you will score 10 points.

If your submitted code is a hodge-podge of copy-pasted spaghetti, you will be penalised.

[3] It would be disingenuous of us to allow you to score any points in the Model Performance section, if your model does not comply with the strict training constraints. It would be unfair to the other students whose models did comply.

It's entirely condoned for you to use AI to help you generate code for this assignment, however, please ensure your code is of an acceptable standard.

If your code is easy to read, and easy to use, you'll make the TAs' lives easier, and you'll be rewarded!

3.3 Project Report

- 70 marks
- 2500 words maximum! (This is not a target number, and we assume reports would be around 3 pages if there were no figures.)
- Figures, plots etc. are all encouraged!
- Must include a screenshot of the SUMMARY from running your Model Inference notebook. This isn't to encourage overfitting. It's just so we can validate that the submitted code didn't change after you wrote your report.
- Must include the seed you used in the Model Training notebook, for that final model.

We weren't kidding when we said this is the most important part of the assignment. Don't forget to include as much evaluation and insight into your model architecture as possible.

This section is where you will be rewarded for your Model Training Code, to the extent that your work in your Model Training Code has permitted you to write up an interesting and creative Project Report. Remember, please keep the analysis to your Project Report. *Generate* the analysis in your Model Training Code, and *present* it to us in one cohesive narrative in your Project Report.[4]

Remember - we are looking for creative and interesting approaches, we don't really care that much about absolute model performance, and visualisations are *always* a good idea.

Best of luck!

[4] We will check your Model Training Code to make sure you have actually done the work, that you've generated the plots yourself, and that they are real, but that should hopefully be as deep as we need to go into your code. We want to rely primarily on your report.