Arnau Pons May 2025
Fluendo's Technical Test

1. **What do you think it occurred during this model development (training & evaluation)?**

   Different issues may come to mind that could have impacted negatively the performance of the model during training and evaluation and led to misreading license plates. Firstly, an imbalanced training data set; the model was likely trained on a limited UK's license plates dataset, potentially overrepresenting certain type of plates. Moreover, the data could have lacked variety, for example similar characters such as "0" and "O", or "V" for "Y" as in the news. This lack of cases may lead to poor results if the case is found later.

   Another problem that can appear is overfitting, which can be normal when using a normal deep learning model in a computer vision task. For the same reason as mentioned before, the model may have been specialized in some type of plates, and when finding new cases in less-likely seen situations, such as lack of visibility, different angles or even blurry images the model does not perform as hoped.

   For the evaluation part, wrong evaluation metrics could have been used. Of course, missing 3 letters out of 6 is worse than just missing one, but in this case missing just one does also signify legal consequences, so precision must be perfect. This also reflects a case of human overlooking, as monitoring some evaluation cases could have easily pointed out that the model was generating errors.

2. **How would you fix this behavior? Please provide at least 2 options explaining their pros and drawbacks**

   The easier answer here is to change the model. We could use first a model that detected where the plate is and then a traditional computer vision method to get the character. The modular design would allow each part to be optimized separately, which can be both a pro and a drawback if not done properly. Moreover, this should work better for any type of plate, so it would have a better generalization and not underperform in strange cases. The main problem this architecture has it is obvious; we now have two models instead of one, which can lead to higher complexity, memory and inference time. Moreover, the dataset should also contain images pointing where the plate is, which is even more expensive than just labeling.

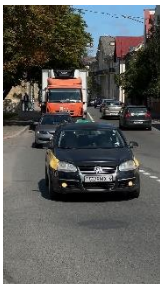
Figure 1: Image

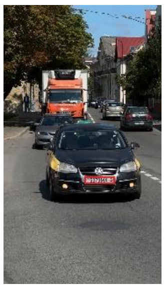
Figure 2: License plate finding

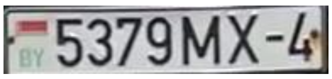

Figure 3: Character identification

Basing on the previous question, one other option could be augmenting the dataset, retrieving a larger and more varied dataset of plates. This will help the model's generalization and will, most likely, boost the model's performance while not modifying the original architecture. The major setback is the high cost, as humans themselves must manually tag each image, so synthetic plate data can do the trick. Another drawback can be that is impossible to keep up to date the model, because new plates are released every day, and it is not feasible to have the model trained with this periodicity.

3. **What do you think it will occur when running this AI in a different country with different plates formats? How would you ensure system accuracy?**

   The model is likely to perform poorly for many reasons; to begin with, each country has its own formats, and when exposing a never seen format before the model may misinterpret them, which is normal considering that similar data cannot be found in training dataset. Another issue that can appear is character misrecognition, other alphabets are not taken into account when training and then a correct output is not possible. Take, for example, a Japanese license plate:

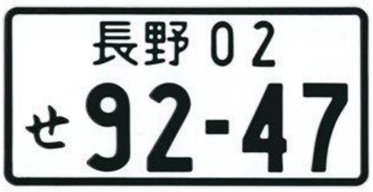

Figure 4: Japanese license plate. Kanjis and hiragana are used.

   To ensure system accuracy it depends on the use we want to give to the algorithm. If the model just needs to identify a particular country's license plate it is enough with just fine-tuning the model using labeled images from the target country.

   This problem gets harder when the model needs to recognize a plate from any country. What could be done here is first run a model that recognizes the country or region where the license plate is from and then run a specific computer vision method, such an OCR, to identify the characters. This way we avoid overfitting over some formats and avoid false negatives, with the extra cost of having more models.

4. **Do you know any OCR (Optical Character Recognition) algorithms (Deep learning based) that could be used here?**

   OCR is a system that convert images of text, such as scanned documents or in our case license plates to machine text. To be honest, I am now coursing the computer vision subject in university, so I'm still not very familiarized with this technology, although I have watched some videos and tried out small notebooks with these models. I have used EasyOCR, a python library with a simple API, and works the following way: Firstly, finds areas in the image that contain text, followed by a **convolutional recurrent neural network** (CRNN) to recognize the characters in the text and a later preprocessing. Using CRNNs is especially useful with short sequences and works in many languages, so I think it would be a good choice for this problem.

I also know the existence of Tesseract, which uses LSTMs, and I am more familiar with, but I still have not used it.

5. **Explain a Computer Vision / Artificial Intelligence project in which you have participated (goals, your role, difficulties you found, how they were solved, …)**

   I took part in many projects throughout the degree, but the most demanding project was a clinical prediction task together with *Hospital Clínic de Barcelona*. We were given clinical data about patients and the goal was to predict a certain trait, which cannot be disclosed. My role was mainly data preprocessing, where I had to work with lots of separate datasets, that contained punctual records. The principal difficulty was putting together all the data; rows had different names for the same observations, so a lot of careful preprocessing needed to be done. There were many columns, more than 200, so feature engineering was a key step, as well as handling missing values, which appeared in big amounts and outliers.

   This was also the biggest project I worked in so far, we were 8 members, so it was the first time we applied agile methodologies, which helped me develop efficient communication and teamwork. The relationship we had with the hospital was like a real client, so it helped me develop real word experience too. Some other problems we experienced were the poor availability of the client, as well as changing requirements, which were solved by careful planning and hard work.