I was tasked with improving the results and scope of the **image classifier network** we were using for **industrial part recognition**. The original idea proved to be too difficult to optimize in the long term, so with a little push from a consulting expert I began to train a new network from scratch, marking my foray into the world of machine learning.

Increasing the difficulty was the fact that the **data quality was sporadic**, **without** a realistic **chance to re-capture any video** footage, and several mislabellings both in the training and testing data, so I had to put a lot of effort into visualizing the performance of each training run to find out what the issues were.

Uncovering these errors in the datasets forced me to engineer various ways to detect and fix them, ultimately creating an **ingestion and processing toolchain** in python, utilizing several steps of **quality assurance** along the way, like custom neural networks trained to detect issues and mark files before handing them off to later processing steps.

The result was a model that proved capable in real-world tests and performed well above the expectations: 84% accuracy in single-image top-4 scenarios, and **perceptually 100% accuracy** as the system was upgraded to enable multi-image prediction.

Another task of mine was the prototyping of an **OCR software** solution, where we had to detect writing from stamped machine parts to hand-engraved dotmatrix serial numbers.

While this project never got out of the prototype phase, the solution I came up with, (along with the pre-processing, hyperparameters, and prospective fine-tuning method) resulted in our version mostly keeping up with, and in some more niche and difficult cases even surpassing industry standard solutions, such as Azure's AI Vision. The fine-tuning solution's prototype was also completed: a **synthetic data generation script** written from scratch for Blender, opening the gateway to fine-tune the model incredibly well for specific scenarios if needed.