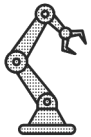
Key learnings

Optimising model inference on CPU

ML team, 2022

Who we are

Hasty is a Berlin startup offering a unified agile ML platform for the entire vision AI pipeline

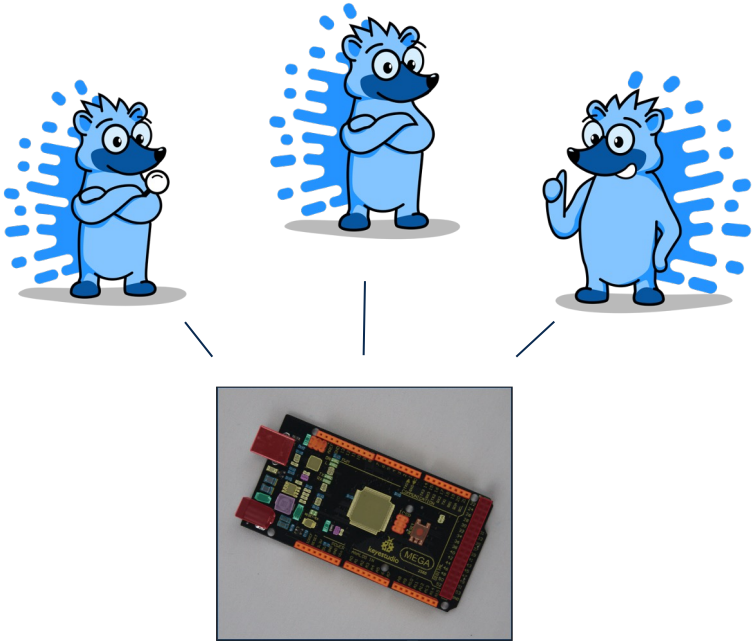


We started as a team of ML engineers building vision AI applications in German manufacturing. **We cut our teeth there** – making us intimate with the challenges of getting to production in vision AI.

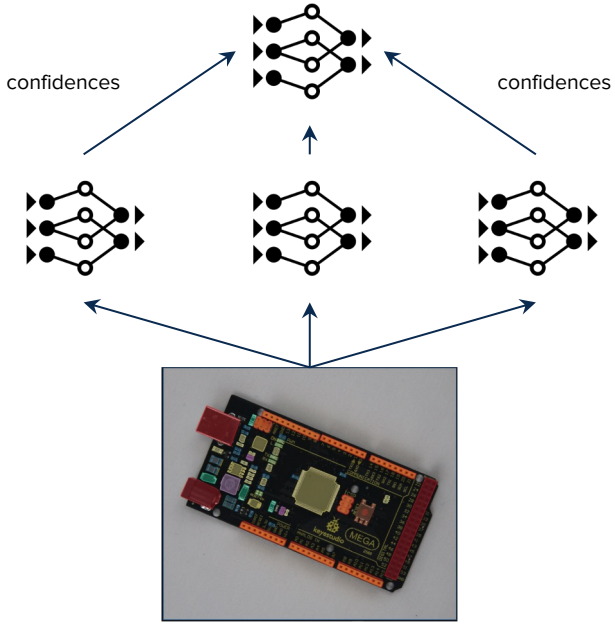
So we started building our tool in **2019** and are venture-backed since **2020**. Our main investors are Shasta Ventures from San Francisco and coparion from Germany.

Today, we are a team of more than 30 and work together with **over 12,000 users** around the globe — from small startups, over research institutions, to large multinationals — and have **trained over 100,000** models.

Improving the quality of the data asset



Standard approach



AI-based approach

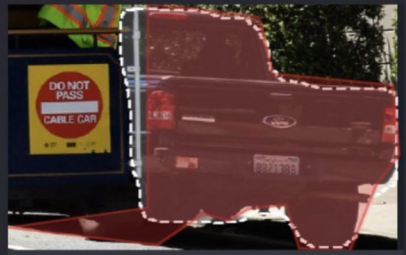
Case study

The tool can deal with four types of errors

Find labels that annotators missed

Error type: Missing label
Predicted: cable car (52.62%)

Find too small or big labels



Error type: Low Iou
Iou: 80.96%

Find artefact annotations

Error type: Extra label

Find labels with the wrong class

Predicted: heritage stree... (53.73%)
Current: cable car (42.73%)

Algorithm

In the heart of the algorithm lies out-of-sample predicted probabilities.

For better performance the **probabilities have to be well-calibrated.**

[Submitted on 31 Oct 2019 (v1), last revised 8 Apr 2021 (this version, v5)]

Confident Learning: Estimating Uncertainty in Dataset Labels

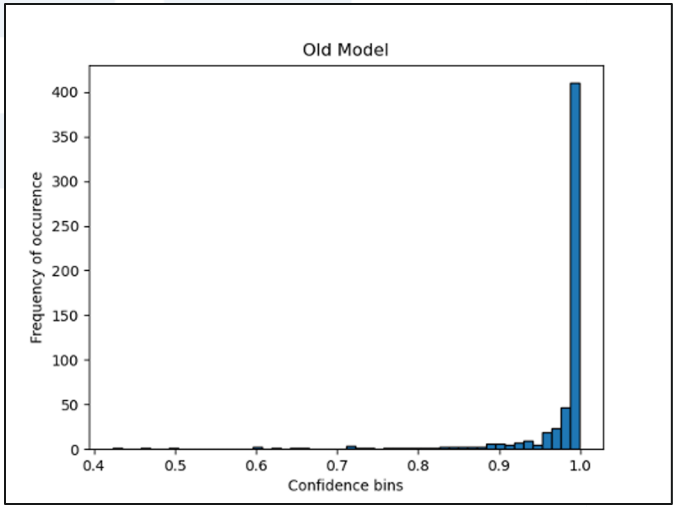
Curtis G. Northcutt, Lu Jiang, Isaac L. Chuang

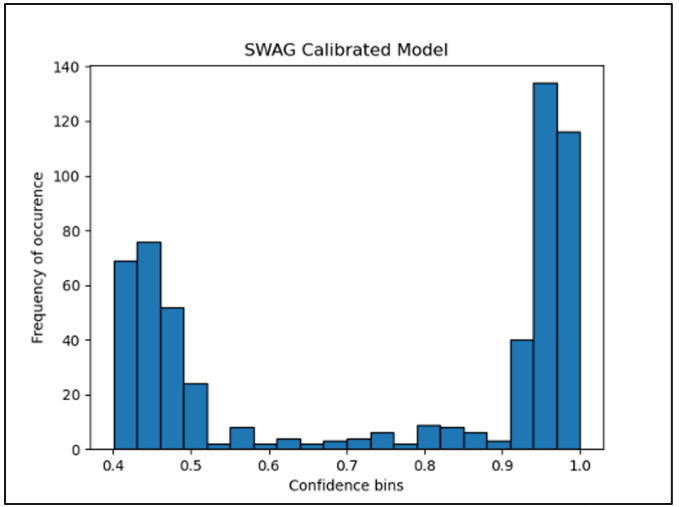
Case study

SWAG proved to be the best approach

We tested several SOTA approaches and **Stochastic Weighted Average Gaussian (SWAG)** proved to be the best one:

- easy to integrate
- showed the best performance
- **great calibration properties**





The problem

To initialize a SWAG model for prediction it is needed to sample from the gaussian distribution with mean as the weight value.

The more models we sample from the SWAG distribution (and use as an ensemble) the more calibrated probabilities will be.

Memory becomes an issue that's why it was decided to use CPU for the inference.

Setup

Tests were run on GCP **n2-highcpu-32** instance

Intel® Extension for PyTorch

We used code optimizations from Intel team to speedup CPU inference

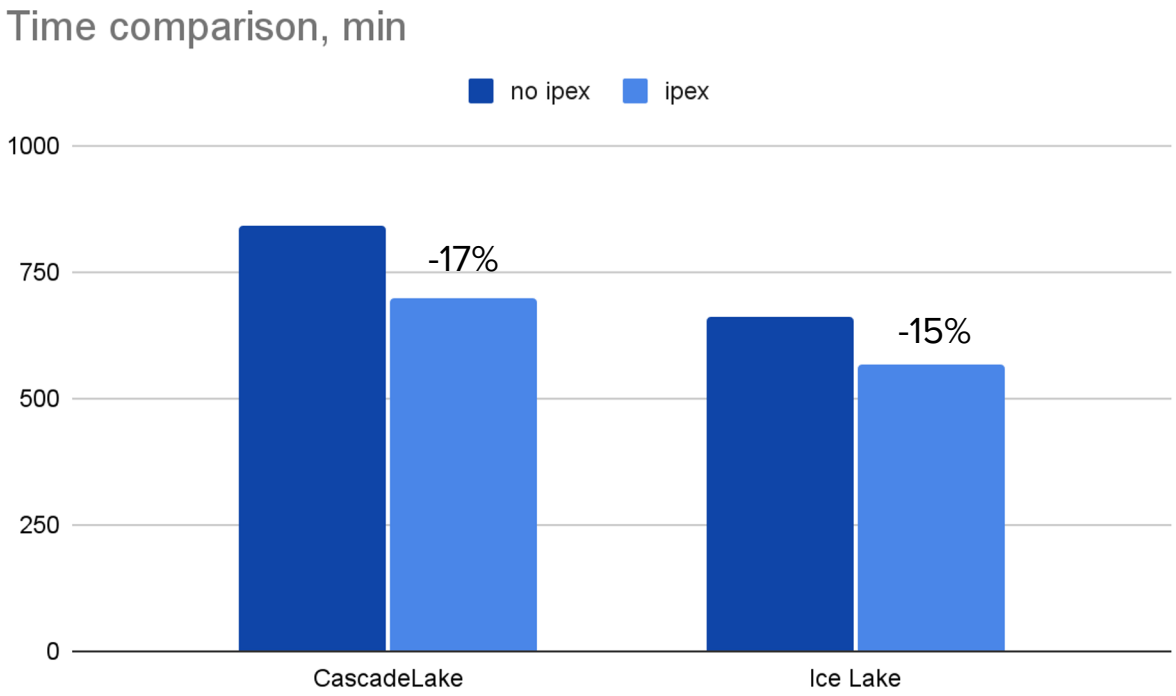
```
import torch
import torchvision.models as models

model = models.resnet50(pretrained=True)
model.eval()
data = torch.rand(1, 3, 224, 224)

import intel_extension_for_pytorch as ipex
model = model.to(memory_format=torch.channels_last)
model = ipex.optimize(model)
data = data.to(memory_format=torch.channels_last)

with torch.no_grad():
  model(data)
```

Results


Time comparison, min
no ipex
ipex
1000
750
500
250
0
-17%
-15%
CascadeLake
Ice Lake

Next steps

Try out Intel optimisations on FP32

Model quantization

Benchmark on FP16, Int8

Thank you