Learning algorithms, in particular deep-learning ones, require the tuning of many meta-parameters.

These parameters have a strong impact on the performance, resulting in a “meta” over-fitting through experiments.

We must be extra careful with performance estimation.

Running 100 times the same experiment on MNIST, with randomized weights, we get:

<table>
<thead>
<tr>
<th>Worst</th>
<th>Median</th>
<th>Best</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.3%</td>
<td>1.0%</td>
<td>0.82%</td>
</tr>
</tbody>
</table>
The ideal development cycle is

```
Write code → Train → Test → Paper
```

or in practice something like

```
Write code → Train → Test → Paper
```

There may be over-fitting, but it does not bias the final performance evaluation.

Unfortunately, it often looks like

```
Write code → Train → Test → Paper
```

⚠️ This should be avoided at all costs. The standard strategy is to have a separate validation set for the tuning.
When data is scarce, one can use cross-validation: average through multiple random splits of the data in a train and a validation sets.

There is no unbiased estimator of the variance of cross-validation valid under all distributions (Bengio and Grandvalet, 2004).

Some data-sets (MNIST!) have been used by thousands of researchers, over millions of experiments, in hundreds of papers.

The global overall process looks more like

![Diagram](image-url)
“Cheating” in machine learning, from bad to “are you kidding?”:

- “Early evaluation stopping”
- meta-parameter (over-)tuning
- data-set selection
- algorithm data-set specific clauses
- seed selection.

Top-tier conferences are demanding regarding experiments, and are biased against “complicated” pipelines.

The community pushes toward accessible implementations, reference data-sets, leader boards, and constant upgrades of benchmarks.

References