Twitter Thread by Russell Kaplan

Russell Kaplan
@russelljkaplan

Lessons learned debugging ML models:

1/ It pays to be paranoid. Bugs can take so long to find that it’s best to be really careful as you go. Add breakpoints to sanity check numpy tensors while you're coding; add visualizations just before your forward pass (it must be right before! otherwise errors will slip in).

2/ It's not enough to be paranoid about code. The majority of issues are actually with the dataset. If you're lucky, the issue is so flagrant that you know something must be wrong after model training or evaluation. But most of the time you won't even notice.

3/ The antidote is obsessive data paranoia. Without this, data issues will silently take away a few percentage points of model accuracy.

4/ You can unit test ML models, but it's different from unit testing code. To prevent bugs from re-occurring, you have to curate scenarios of interest, then turn them into many small test sets ("unit tests") instead of one large one.

5/ Each unit test should have an evaluation metric of interest, a pass/fail threshold, and be defined for a curated subset of data.

6/ Without model unit tests, you will see aggregate metrics improving in your evaluation but introduce critical regressions when you actually ship the model. Unit tests are a requirement to durably fix bugs in ML-powered products.

7/ Bugs are fixed faster when iteration times are faster. Impose a hard ceiling on model training time, even if you could squeeze more gain by training a bit longer. In the long run, experiment velocity >> performance of one model.