

# A Brute Force Approach: Forcing the Network to See Invariances

- Convolutional Networks architectures use knowledge about invariances to design the network architecture/weight constraints
- But it's much simpler to incorporate knowledge of invariances by just creating extra training data:
  - for each training image, produce new training data by applying all of the transformations we want to be insensitive to (Le Net can benefit from this too)
  - Then train a large, dumb net on a fast computer.
  - This works surprisingly well if the transformations are not too big

# Preventing overfitting: Ways to make predictors differ

- Rely on the learning algorithm getting stuck in a different local optimum on each run.
  - A dubious hack unworthy of a true computer scientist (but definitely worth a try).
- Use lots of different kinds of models:
  - Different architectures
  - Different learning algorithms.
- Use different training data for each model:
  - **Bagging**: Resample (with replacement) from the training set: a,b,c,d,e -> a c c d d
  - **Boosting**: Fit models one at a time. Re-weight each training case by how badly it is predicted by the models already fitted.
    - This makes efficient use of computer time because it does not bother to “back-fit” models that were fitted earlier.