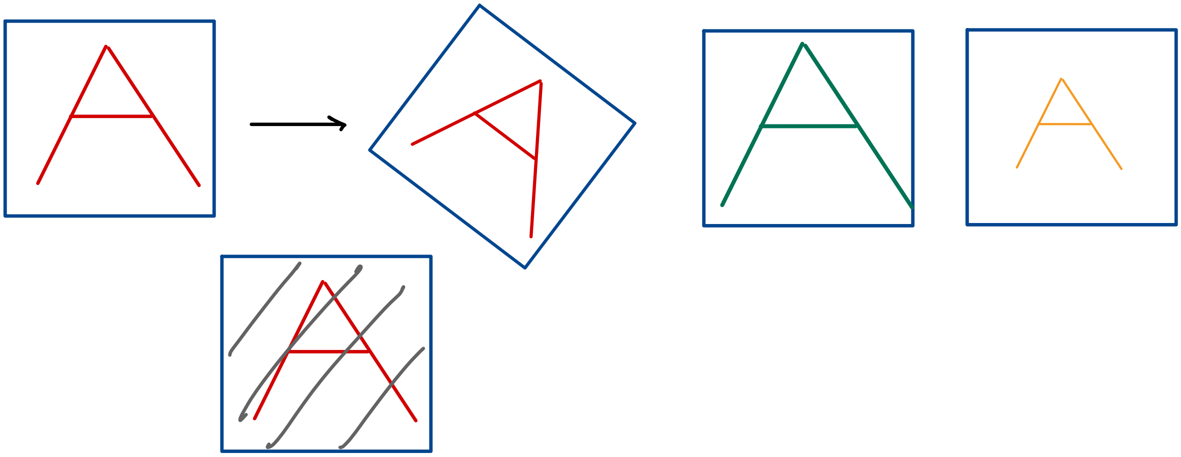
Adding data for areas which need more help as indicated by error analysis would be much better compared to adding data for everything as that can be much slower and computationally more expensive unless we have some ways through which we can add "more data for everything" easily.

Still there are multiple ways through which we can add data

Data Augmentation

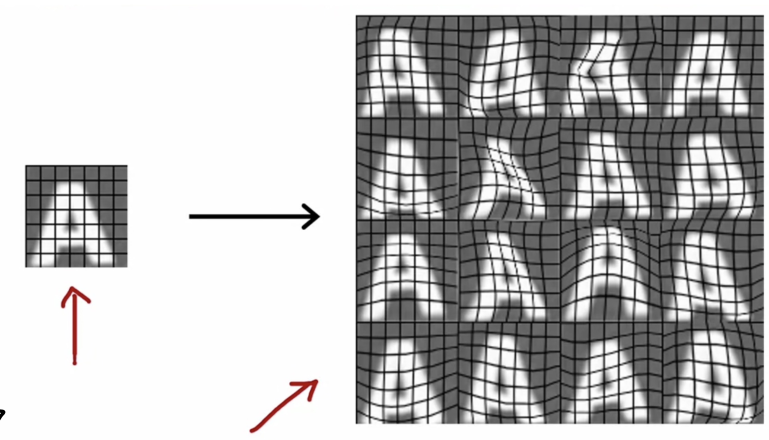
→ modifying existing training examples to create new training examples.

Suppose, we have an image of the letter A.



These are ways of rotating, scaling, creating distortions in the image, in some cases mirroring the image. These techniques don't change the fact that the letter is still A i.e. the original identity of the image.

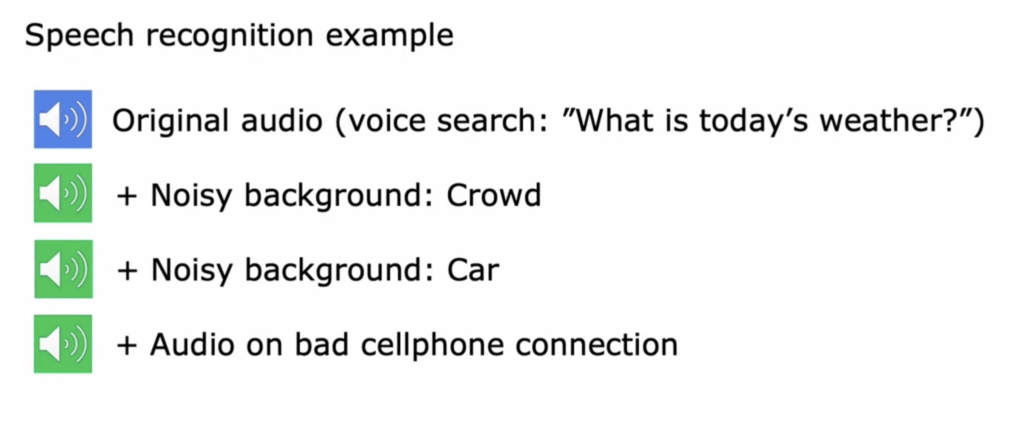
This helps us create more training examples from one example thus helping us add more data.



You can also create distortion and create a whole new library of examples which you can feed the algorithm.

Data augmentation also works for audio.

In speech recognition models you can change the original audio by adding different noisy backgrounds.



Speech recognition example

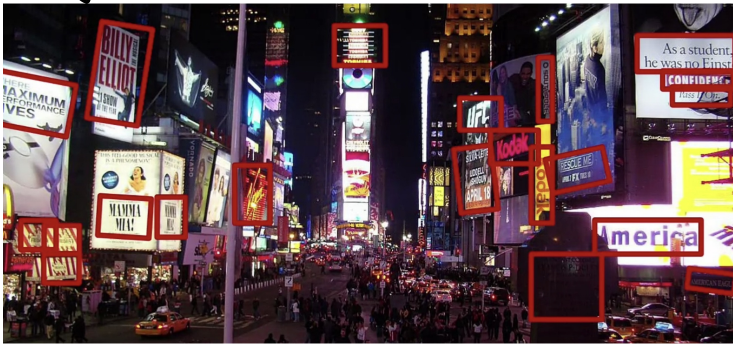
- Original audio (voice search: "What is today's weather?")
- + Noisy background: Crowd
- + Noisy background: Car
- + Audio on bad cellphone connection

# One very important thing is that meaningless changes don't help your model.

eg. Adding pixel noise to the letter A won't do anything. We should make changes which are different but in a way correspond to our original input.

Data Synthesis

→ Use artificial data inputs to create a new training example.



Above image is being used for computer OCR. In this case, the model needs to read text from an image like this.

Real Data
↓
task of the model and make out what the word in the middle is.

Synthetic Data
↓
Created from using diff. fonts, colours, etc. on a text editor. Using this technique we can generate very large datasets for OCR tasks.

Different engineering paradigms

Traditional approach → due to this most algorithms are good enough

↓

Conventional "model centric" approach → $AI = \boxed{Code} + Data$

focus on improving the code.

adopting more of a data-centric approach.

"Data centric approach" → $AI = Code + \boxed{Data}$

Focus on increasing quality of data.