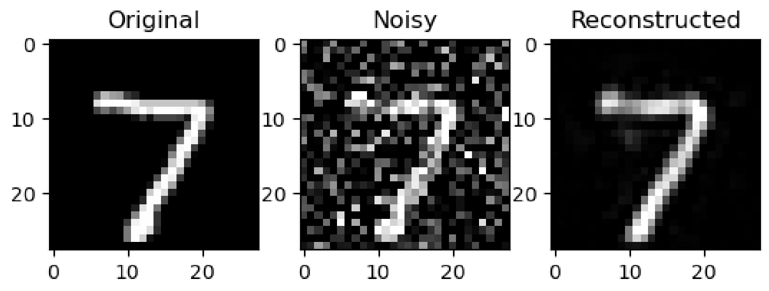

Figure 1: Example for 1 sample of the test dataset. Report 10 such examples from the test dataset

3 Encoder-Decoder with Attention [16 marks]

Implement the following paper: data2vis. The train, dev, and test splits are provided as train.sources, train.targets, dev.sources, dev.targets, and test.sources, test.targets. A dataset named progression.json has been provided for inference.

3.1 Architecture

The following architecture was used by the paper, you can modify it if desired.

- Embedding dimension: 512.
- Encoder: Bidirectional LSTM, 2 layers, 512 hidden units.
- Decoder: LSTM, 2 layers, 512 hidden units.
- Droupout: probability 0.5 before every encoder/decoder layer, only on the inputs, not the hidden/cell states.
- Attention type: Dot product.

- Attention vector dimension: 512.
- Max source sequence length: 500 (first 500).
- Max target sequence length: 500 (first 500).
- Max decode sequence length: 2000 (during inference, first 2000).
- Width for beam search: 15 (beam search is only used during inference).
- Optimizer: Adam, lr=1e-4
- Batch size: 32
- Number of steps: 20000 steps (not epochs, this means a total of 20000 mini-batches were used for training).

3.2 Evaluation

Marks will be provided based on the code but the model should still perform decently, it doesn't need to be perfect. The following things are expected:

- Average log perplexity of 0.032 on the test dataset.
- Generate 15 different possible encoding strategies for progression.json and save them.
- Visualize and save all of the encodings using online editor. If some of the produced encodings have mistakes, you can correct them manually for visualization purposes.

An example vegalite specification named example_vegalite.json has been provided as well. The data has been loaded already. The only thing you need to do is replace the provided mark and encoding with your model's outputs.