**Which setup authors consider?**

The MQ-RNN architecture resembles the Seq2Seq with context. A vanilla LSTM to encode all history into hidden states. Instead of using an LSTM as the recursive decoder, MQ-RNN has a design of two MLP branches. The first (global) MLP summarizes the encoder output plus all future inputs into two contexts. The second (local) MLP applies to each specific horizon. It combines the corresponding future input and the two contexts from the global MLP described earlier, then outputs all the required quantiles for that specific future time step

**Which types of time series / from which domain the architecture is designed for?**

The approach accommodates both temporal and static covariates for multivariate time series.

The model is applied to two domains:

1. Demand forecasting problem at Amazon
2. Global Energy Forecasting - electricity load and electricity price

**What are the distinctive characteristics of those time series if any?**

1. Demand forecasting at Amazon - Available covariates include a range of suitably chosen and standard demand drivers in three categories: history only, e.g. past demand; history and future, e.g. promotions; and **static**, e.g. product catalog fields.
2. Electricity forecasting - the quantity to forecast is a single series of hour-grain price or load from several years and thus there is no static series-related information.

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
How does this paper extend on the prior art?**

This paper method differs from DeepAR by using the more practically relevant Multi-Horizon strategy, a more efficient training strategy and directly generating accurate quantiles.

**What additional benefits does it bring to the domain?**

1. An efficient training scheme for the combination of sequential neural nets and Multi-Horizon forecast - **forking-sequences** which can improve training stability and performance of encoder-decoder style recurrent nets or ConvNets, by training on all time points where a forecast would be created, in a one pass over the data series.
2. A network sub-structure to accommodate a previously little-attended issue: how to account for known future information, including the alignment of shifting seasonality and known events that cause large spikes and dips.

**Which problem(-s) authors try to solve?**

The MQ-R(C)NN model is designed to solve the large-scale time series regression problem. Each series is considered as one sample fed into a single RNN or CNN, even if they correspond to different items. This enables cross-series learning and cold-start forecasting for items with limited history.

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
What is the main idea or ideas authors use to solve the problem outlined?**

The core design of MQ-RNN is using a series of multi-horizon, future-aligned forked decoders that forecast quantiles.

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
What are the technical aspects of the architecture?**

The proposed framework creates Multi-Horizon forecasts by placing a series of decoders, with shared parameters, at each recurrent layer (time point) in the encoder, and computes the loss against the corresponding targets (future series relative to that time point; can be populated on-the-fly in implementation). Thus, planting the nature of forecasting-at-each-time application structurally into the neural net training. Then one backpropagation-through-time can gather the multi-horizon error gradients of different FCTs in one pass over a sample, with little additional cost.

**How hard is it to train?**

As a result of forking-sequences, each time series of arbitrary length serves as a single sample in our model training, eliminating the need of data augmentation, and dramatically reducing the training time.

**How much data does it require?**

This training scheme boosts model performance and regularizes learning stability by efficiently using all information in one shot, while previous algorithms need to cut and down-sample data.

**Do authors motivate the choice of hyperparameters, specifics of the architecture (number of layers or hidden units, etc.), and so on?**

The horizon-agnostic context is included in the model, based on the idea that not all relevant information is time-sensitive.The researchers suggest that adding this structure to the model improves the stability of learning and the smoothness of generated forecasts. The MLP can be removed for the cases where there is no meaningful future information, or sharp and spiky forecasts is not desired.

Encoder extension – feeding past series as lagged feature inputs, along with yt, into the recurrent layer at t. Hence, effectively constructing skip-connections to past values of the input series before passing them through the recurrent layer, as opposed to having skip-connections after RNN. Therefore, improving vanilla LSTM performance.

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**How does architecture perform compared to prior art?**

MQ\_RNN has consistently the best accuracy across all horizons. The training loss curve of MQ \_RNN\_cut is more volatile and flattens out early. Series-level diagnostics also indicate similar high-level behaviours between MQ\_RNN and MQ\_RNN\_cut, but the latter has worse performance. In terms of calibration ML\_RNN is slightly overbiased, and its 80% prediction interval is on average almost twice as wide as MQ\_RNN. By extending the sequential encoder beyond vanilla LSTM, further accuracy gain is achieved within the proposed MQ-RNN framework. MQ\_RNN\_lag is the best RNN-type model, while MQ\_CNN\_wave has the highest accuracy overall, both with otherwise similar forecast behaviour.

**Elaborate on both metrics and conceptual benefits of the model (being able to predict sudden spikes, or producing probabilistic forecasts, etc.).**

As shown below, the model handles successfully new product cold-start situation as well as promotional spikes:

****

The local MLP is the key to aligning future seasonality and events and the capability to generate sharp spiky forecasts.

For the electricity price forecasting problem, the average quantile loss of MQ-RNN was **2.63**

For the electricity load forecasting problem, the average quantile loss of MQ-RNN was **7.43**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Estimate efforts needed to create a prototype of the architecture (without packing it into production-grade wrappers).**

In order to build a simple prototype, we need to do the following:

1. Create Encoder class which has a traditional seq2seq model and is based on LSTM.
2. Create both GlobalDecoder and LocalDecoder classes for the Decoder part
3. Create MQ\_RNN class which uses the encode/decoder classes and implements the train / predict / loss functions.
4. Implement the dataloader
5. Get relevant data to work with

**Which short-cuts you can take to create a prototype (smaller model, less data, different dataset, etc.)?**

The encoder can be pure vanilla LSTM with no extensions.

We can cave dataset with minimal history records and get ground truth results by predicting future results and feeding them to the network.

In order to capture longer time dynamics without too long RNNs, we can run the encoder at a daily grain, keeping the forecasting decoder grain as hours.

The network can be not so intensively tuned. The final setting is based on intuitive tries. The major parameter choices are the duration of the time-steps that the RNN is modeling (number of recurrent layers) and the number of RNN states.

We can use the lag-series trick as MQ\_RNN\_lag in the Amazon problem - feed past series as lagged feature inputs, along with yt, into the recurrent layer at t.

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**How would you employ this model in your domain? would it fit into it or improve over the approaches you already use?**

Working on an NLP project for multiclass entity extraction. Hence, a Multi-Horizon Quantile Recurrent Forecaster is not relevant for time series, though we don’t mind to use it 😊