CAMul: Calibrated and Accurate Multi-view Time-Series Forecasting

Harshavardhan Kamarthi, Lingkai Kong, Alexander Rodriguez, Chao Zhang, B. Aditya Prakash
College of Computing, Georgia Institute of Technology, USA
{harsha.pk,lkkong,arodriguez,chaozhang,badityap}@gatech.edu

ABSTRACT
Probabilistic time-series forecasting enables reliable decision making across many domains. Most forecasting problems have diverse sources of data containing multiple modalities and structures. Leveraging information from these data sources for accurate and well-calibrated forecasts is an important but challenging problem. Most previous works on multi-view time-series forecasting aggregate features from each data view by simple summation or concatenation and do not explicitly model uncertainty for each data view. We propose a general probabilistic multi-view forecasting framework CAMul, which can learn representations and uncertainty from diverse data sources. It integrates the information and uncertainty from each data view in a dynamic context-specific manner, assigning more importance to useful views to model a well-calibrated forecast distribution. We use CAMul for multiple domains with varied sources and modalities and show that CAMul outperforms other state-of-art probabilistic forecasting models by over 25% in accuracy and calibration.

CCS CONCEPTS
• Computing methodologies → Machine learning. • Information systems → Data mining; Web mining.

KEYWORDS
Multi-source multi-modal data, Probabilistic forecasting, Uncertainty quantification, Time-series Forecasting

ACM Reference Format:
Harshavardhan Kamarthi, Lingkai Kong, Alexander Rodriguez, Chao Zhang, B. Aditya Prakash. 2022. CAMul: Calibrated and Accurate Multi-view Time-Series Forecasting. In Proceedings of the ACM Web Conference 2022 (WWW ’22), April 25–29, 2022, Virtual Event, Lyon, France. ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3485447.3512037

1 INTRODUCTION
Time-series forecasting is a classic machine learning problem with applications covering wide-ranging domains including retail, meteorology, economics, epidemiology and energy. For many of these applications, we have a wide variety of datasets representing different views or perspectives of the phenomena to forecast. These views may differ in their structure and modality, and also in the quality and reliability due to collection and processing differences. Due to the high-stakes decisions that these forecasts may inform (e.g., hospital resource allocation for COVID-19, planning energy infrastructure for cities), designing ML models that can leverage these multiple data sources to provide not only accurate but also well-calibrated forecast distributions is an important task.

State-of-the-art time series forecasting methods employ sequential neural models that can naturally integrate multiple time-varying exogenous features [35, 42, 47]. However, they are not suitable to ingest multiple modalities of data such as networks, sequences and fixed sized features together. Further, effective integration of information from multiple views is challenging because information from individual views may be noisy, conflicting, redundant or sometimes unreliable. Recent works that seek to integrate multi-source and multi-modal data for accurate sequence forecasting by using architectures like Graph Neural networks and Convolutional networks (such as for disease [41] and sales forecasting [11], multimodal sentiment analysis [53]) do not comprehensively address this challenge. They propose to combine embeddings from multiple data sources either through direct concatenation [19, 50, 56] or simple summation [41]. Other recent methods [5] learn a view-specific importance parameter and use it to combine embeddings in a weighted manner. However, these weights are learned over all data points and are not context-sensitive: they do not account for temporal variation in the importance of each view and variance across time series.

Moreover, these multi-modal time-series forecasting methods do not focus on learning a well-calibrated forecast distribution. This is especially challenging in a multi-view setting since the ambiguity and reliability of views has to be integrated to better inform forecast uncertainty. Redundancy and high confidence in beliefs across multiple data views could inform a more confident
forecast whereas conflicting information and lower confidence from some data sources may suggest higher uncertainty. Past works on multi-view probabilistic forecasting [54] do not integrate stochastic information about beliefs from individual data views and instead use a deterministic embedding from fusion of view-specific embeddings to learn distribution parameters. As a result, these methods do not model the view-specific uncertainty which may result in mis-calibrated forecasts.

Hence in this paper, our work tackles the challenge of modelling as well as integrating information and uncertainty from each data view to provide accurate and well-calibrated time-series forecasts (Figure 1). We introduce a general multi-view probabilistic time-series forecasting framework, CAMul (Calibrated and Accurate Multi-view Forecasting), that jointly models uncertainty from multiple views independently using a latent embedding distribution. It then combines the views in a context-sensitive mechanism by accounting for their reliability specific to the given sequence, thus providing well-calibrated and accurate forecasts. To learn a view-specific distribution capturing relevant information for each data source, we use a non-parametric modelling framework. We directly use latent embeddings of the data points in the training set of each view in the functional space to allow flexible representation of the predictive distribution that rely on similar patterns seen before. Our main contributions are:

a) Probabilistic Neural Framework jointly modeling multi-view uncertainty: We propose a general framework CAMul for probabilistic time series forecasting on multi-modal and multi-source data making no assumptions on the structure of data. Our non-parametric probabilistic model leverages probabilistic relations learned between latent representations of data points for each data view to account for uncertainty from each view.

b) Integrating multi-view uncertainty towards calibrated forecasts: CAMul leverages the latent information and uncertainty from each view and carefully integrates the beliefs from multiple views together, dynamically weighting each view’s importance based on input data, to learn well-calibrated predictive distribution.

c) Evaluation of CAMul on multiple domains: We use the CAMul framework to design models for multi-view time-series forecasting tasks from different domains using diverse data sources and modalities (static features, sequences, networks). We compare CAMul against state-of-art domain-specific as well as general forecasting baselines and show that CAMul models clearly outperform all baselines by over 25% in accuracy and calibration. We also show both empirically and using case studies, that our method of modeling and integrating uncertainty from individual data sources indeed causes these improvements.

2 RELATED WORK

Probabilistic Time-series Forecasting Classical time-series forecasting like exponential smoothing and ARIMA-based models [21] focus on univariate time-series with a small number of exogenous features and learn model parameters independently for each sequence. Recent probabilistic forecasting models leverage the representation power of neural sequential modules like DeepAR [47] which directly models the mean and variance parameters of the forecast distribution, Bayesian Neural Networks [57] which require assigning useful priors to parameters and require high computational resources for learning. Some recent works inspired from the space-state models explicitly model the transition and emission components with deep learning modules such as in the case of Deep Markov Models [29], and recent Deep State Space models [35, 42]. Others introduce stochasticity into state dynamics of recurrent neural networks such as Stochastic RNN [12], Variational RNN [8] and State Space LSTM [55]. Neural Process (NP) [14] models a global latent variable for entire dataset to capture uncertainty with is used with input data’s embedding to model the distribution parameters. Recurrent Neural Process [40] leverages NP for sequence data. Functional Neural Process (FNP) captures stochastic correlations between input data and datapoints from training distribution to provide a flexible non-parametric mechanism to model output distribution using related training data points in functional space. EpiFNP [24] a state-of-art calibrated disease forecasting model, is closest to our work and leverages Functional Neural Process (FNP) [38], which uses stochastic correlations between input data and datapoints to model a flexible non-parametric distribution for univariate sequences. Our work leverages FNP for uncertainty modeling of each of the individual views before we jointly model the forecast distribution combining distributions from different views. Multi-view time-series forecasting Recent advances in deep learning architectures has allowed us to extract representations from variety of data sources such as images [36], sequences [46], graphs [16], text [10], etc., and combine these modules’ representation for training in a end-to-end fashion. In order to integrate these representations, most methods employ simple summation or concatenation methods [31, 37, 50, 54, 56] either at the initial layers of model (early fusion) or last layers (late fusion) [13, 32, 39, 53]. For instance DeepCovidNet [41] uses spatio-temporal and static features using simple summation for Covid-19 prediction. Ekaram et al. [11] similarly integrate images, text and static features for predicting consumer sales. Moreover, most of these methods do not focus on probabilistic forecasting unlike [5] which uses EHR sequence data, static demographic and location data and learn a data-source specific weight by pre-training the aggregated embeddings on prediction task of deciding if patient requires a treatment. Then, they employ a Gaussian Process approach to learn forecast probability on these embeddings to capture uncertainty. In contrast, we model data-view specific uncertainty by learning a latent distribution for each view which allows the model to reason about source specific uncertainty as it integrates uncertainty-aware stochastic representations of all views towards the forecast distribution.

3 PROBLEM FORMULATION

We describe the Multi-view probabilistic forecasting problem, a generalization of the probabilistic forecasting problem to multimodal and multi-source data. We denote the time-series dataset as $\mathcal{Y} = \{Y_i\}_{i=1}^N$ where individual time-series is a univariate sequence, $Y_i = \{y(t) : y(t) \in R, 1 \leq t \leq T\}$. At each time-step $t$, we also have exogenous data sources that can be used to forecast future values in $\mathcal{Y}$. We divide these features from multiple sources and modalities into views. Assume we have $K$ views for a specific problem representing $K$ sources/modalities of data. The dataset for a view $j$ is denoted by $\mathcal{X}^{(j)} = \{X_i^{(j)}\}_{i=0}^N$. We do not assume any
We first give an overview of whose values do not change across time: static views \[24\], and then describe each of its specific data type for each view. However, all datapoints of a view are assumed to have the same type. We can also have static views whose values do not change across time: \(Y_t\). In some cases, we also denote as \(X[1 : t]_i^{(j)}\), the set of data till week \(t\): \(X[1 : t]_i^{(j)} = \{x[t']_i^{(j)} : 0 \leq t' \leq t\}\).

Given the time-series values \(Y[1 : t]_i\) and exogenous features from all views \(X[1 : t]_i^{(j)}\) till current time \(t\), our goal is to model a well-calibrated forecast distribution

\[
P(y[t + \tau]_{i}|[Y[1 : t]_i]\{X[1 : t]_i^{(j)}\}_{i=1...N,j=1...K})
\]

for forecasting time-series values \(y[t + \tau]_i\), \(\tau\) steps in future. We also assume that the view \(i\) is the sequence dataset, i.e, \(V_i = X_i^{(1)}\) and refer to it as the default view.

4 METHODOLOGY

We first give an overview of CAMul and then describe each of its modules in more detail. Model Overview: CAMul tackles the challenges related to jointly modeling useful information and uncertainty from multiple data-sources. It jointly models the predictive uncertainty by integrating information and uncertainty for each view. It dynamically considers the importance of each view based on input sequence to leverage the more important views and learn the output distribution. We use the functional neural process (FNP) [38] framework to learn a non-parametric latent embedding distribution for each view which enables a flexible method to model complex distribution by leveraging latent similarity with past training data-points from each data view. In contrast to related approaches that use FNP for univariate sequences [24], CAMul solves a more general and harder problem of leveraging stochastic uncertainty from multi-source, multi-modal data and tackles the challenge of jointly modeling and integrating stochastic representations from each view.

At a high level, CAMul accomplishes our goals by performing the following steps in an end-to-end manner: A) learn latent representations for each view using Multi-view Latent Probabilistic Encoders to capture information and uncertainty from each data source, B) leverage stochastic similarity between data-points in latent space to learn relations between time-series for each view via View Specific Correlation Graph and leverage these relations to derive a view-aware embedding distribution, C) integrate information and uncertainty captured by latent embedding distributions from multiple views using learned importance of each view based on input sequence via Context-sensitive Dynamic View Selection Module to derive the output distribution. Figure 2 shows the full pipeline.

For each view \(j\), the set of data-points on which we learn the stochastic similarity w.r.t input point is a subset of the training set \(X^{(j)}\) of the view \(j\) that comprehensively represents the entire training data. This set is called the reference set, denoted by \(R^{(j)}\). For example, for a view that represents sequence data from the past, we consider each of the complete time-series sequences \(X[1 : t]_i^{(j)}\) from past training data as a single data-point in the reference set. Similarly consider views that represent static time-specific data such as months. Each training point for a given month contains the same static features. Therefore, the reference set contains a datapoint for each unique month. Note that, during inference, we retain these reference sets and use them to derive the output distribution.

4.1 Multi-view Probabilistic Latent Encoders

To capture useful information and uncertainty from each view, we learn a latent embedding distribution for datapoints of a specific view. We leverage appropriate neural encoders for each view based on the modality to learn latent distribution parameters. The probabilistic encoder \(g_{\theta_j}\) for each view \(j\) derives the parameters...
of latent embedding distribution for each of the reference points $R^{(j)} = \{r_k^{(j)}\}_{k=1}^{N_j}$ and input training point $x_i^{(j)}$ parameterized by a Gaussian Distribution:

$$
\mu_k^{(j)}, \sigma_k^{(j)} = g_\theta_j(R_k^{(j)}), \quad z_k^{(j)} \sim \mathcal{N}(\mu_k^{(j)}, \sigma_k^{(j)}).
$$

$g_\theta_j$, a neural network module parameterized by $\theta_j$, is chosen based on the modality of view $j$. For example, we use GRU for views with sequence data, Graph Convolutional Network [28] to encode relational data and Feed-Forward networks for static fixed-sized features. The Supplementary contains detailed discussions of encoder architectures for different views used in this paper. The set of all latent encodings of reference set $R^{(j)}$ and training set $X^{(j)}$ is denoted as $Z^{R^{(j)}}$ and $Z^{X^{(j)}}$ respectively.

### 4.2 View Specific Correlation Graph

The latent embedding for input data $z_i^{(j)} \in Z^{X^{(j)}}$ captures the probabilistic information of input data in view $j$ whereas latent variables of $Z^{R^{(j)}}$ capture stochastic information from entirety of the data view $j$. To capture the information from the entire data view $j$ conditioned on datapoint $i$, we use the non-parametric FNP framework that allows modeling a flexible data-view aware latent distribution for given input data by leveraging related reference points of the view.

We first learn probabilistic relations between input datapoint and reference points in latent space using a similarity metric to create a View Specific Correlation Graph (VSCG) and then use the reference points connected to input’s embedding in VSCG to learn the view-aware latent embedding.

We choose the radial basis function (RBF) as the similarity metric to relate reference set $R^{(j)}$ to training set $X^{(j)}$:

$$
k(z_i^{(j)}, z_k^{(j)}) = \exp(-\rho ||z_i^{(j)} - z_k^{(j)}||^2).
$$

for all $z_i^{(j)} \in Z^{R^{(j)}}$ and $z_k^{(j)} \in Z^{X^{(j)}}$.

Next, we sample the VSCG $G^{(j)}$ as a bipartite graph between $R^{(j)}$ and $X^{(j)}$, modeling each edge as a Bernoulli distribution parameterized by RBF similarity

$$p((i,k) \in G^{(j)}) = \text{Bern}(k(z_i^{(j)}, z_k^{(j)})).
$$

Note that during training, we approximate sampling from the discrete Bernoulli using a Gumbel-softmax distribution [22]. Finally, we aggregate the sampled neighbouring reference points $N(i)$ for each training point $i$ to construct the view-aware latent variable $u_i^{(j)}$ (refer Figure 2) as:

$$
u_i^{(j)}, \sigma_i^{(j)} = \sum_{k \in N(i)} \Phi_i^{(j)}(z_k^{(j)}), \Phi_i^{(j)}(z_k^{(j)})
$$

$$u_i^{(j)} \sim \mathcal{N}(\mu(u_i^{(j)}), \exp(\sigma(u_i^{(j)}))).
$$

where $\Phi_i^{(j)}$ and $\Phi_i^{(j)}$ are linear layers. Thus, the view-aware embedding’s latent distribution is derived from sampled reference points to capture view-specific information. The stochastic process of VSCG also models view-specific uncertainty.

### 4.3 Context-Specific Dynamic Views Selection

Not all views are equally useful to learn a well-calibrated forecast distribution. The importance of each view for predictive distribution also varies for each time-series and across time. For instance, for disease prediction, views representing features related to time (like months) may be useful for predicting seasonal changes but its importance may decrease during highly volatile and uncertain weeks near the peak where short-term sequence history and real-time exogenous features are more important. The importance of a view may also change dynamically when the features are corrupted for a small period of time during collection or measurement.

Thus, we need a dynamic mechanism to weigh the importance of each view based on the input sequence. We propose the View Selection module to learn importance weights for each view conditioned on input sequence and then aggregate embeddings from multiple views in proportion to learned weights. Given the view-aware latent variables of all $K$ views for each datapoint $i$ as $\{u_i^{(j)}\}_{j=1}^{K}$, we combine these multiple views’ knowledge towards the construction of final functional for predictive distribution. We leverage the cross-attention mechanism [52] to learn the importance of each view but also condition the weights on input sequence’s representation from default view 1 that encodes the time-series sequence $z_i^{(1)}$ as:

$$
\{a_i^{(j)}\}_{j=1}^{K} = \text{Softmax}_{j}(h_1(z_i^{(1)})^T h_2(u_i^{(j)}))_{j=1}^{K},
$$

where $h_1$ and $h_2$ are linear layers to transform both embeddings to same dimensions. $\{a_i^{(j)}\}_{j=1}^{K}$ denotes the importance of all views for sequence $i$. Finally, we use the weights to combine the latent representations for each view as

$$
\tilde{u}_i = \sum_{j=1}^{K} a_i^{(j)} u_i^{(j)},
$$

Thus, $\tilde{u}_i$, called the combined view embedding represents combined knowledge from correlations learned by all views weighted by their importance for final prediction distribution. We denote the set of combined view embeddings for all sequences as $\mathcal{U} = \{\tilde{u}_i\}_{i=1}^{N}$.

### 4.4 Forecast Distribution Decoder

Having extracted stochastic representations from each of the views leveraging latent encoders, VSCGs and the View selection module, we learn the output distribution via the Decoder module. The decoder module uses the combined view embeddings $\mathcal{U}$ and the sequence embedding of default view $Z^{X^{(1)}}$ to learn the parameters of output distribution. While $\mathcal{U}$ directly uses only the selectively aggregated embeddings of reference sets, $Z^{X^{(1)}}$ leverages local historical patterns of input sequence including novel information that can be used to extrapolate beyond information from reference sets of multiple views. The final decoder process is described as:

$$
e_i = z_i^{(1)} \oplus \tilde{u}_i,
$$

$$
\mu(y_i), \sigma(y_i) = d_1(e_i), \exp(d_2(e_i))
$$

$$
\hat{y}_i^{(t+\tau)} \sim \mathcal{N}(\mu(y_i), \sigma(y_i)),
$$

where $d_1$ and $d_2$ and feed-forward layers and $\oplus$ is the concatenation operator. Here, we used a multivariate Gaussian distribution to parameterize $P(y_i^{(t+\tau)} | e_i)$ since our target sequence has real numbers.
However, this framework can be extended to discrete and integer-valued output by carefully choosing the appropriate distribution and their relevant statistic to predict.

4.5 Model Training and Inference

The generative process of CAMul can be summarized as

\[
P(y_t^{t+T}) = \sum_{R} P(R) \prod_{j \in V} P(y_t^{t+T} | \{ \mathbf{u} \}_{j=1}^J, \mathbf{U}, \mathbf{Z})
\]

\[
P(y_t^{t+T} | \{ \mathbf{u} \}_{j=1}^J, \mathbf{U}) = \sum_{R} P(R) \prod_{j \in V} P(y_t^{t+T} | \mathbf{u}_j, \mathbf{U}, \mathbf{Z})
\]

where \( \mathbf{U} \) is the set of exogenous features and \( \mathbf{Z} \) is the set of latent variables.

Our objective is to increase the log-likelihood of above equation which is intractable due to integrals over real-valued random variables. Therefore, we use variational inference by approximating the posterior \( \prod_{j} P(\mathbf{Z}_j | \mathbf{U}_j, \mathbf{Z}, \mathbf{R}) \) with the variational distribution:

\[q_j(\mathbf{u}_j, \mathbf{z}_j, \mathbf{g}_j | \mathbf{x}_t) = P(\mathbf{z}_j | \mathbf{x}_t)P(\mathbf{g}_j | \mathbf{r}_j)P(\mathbf{u}_j | \{ \mathbf{u} \}_{j=1}^J).
\]

for each view \( j \), where \( q_j \) is a 2-layer network that parameterizes the Gaussian distribution \( q_j(\mathbf{u}_j | \mathbf{x}_t) \) with input sequences \( \mathbf{x}_t \). The ELBO is derived to be:

\[
\mathcal{L} = -E_{q_j(\mathbf{u}_j, \mathbf{g}_j, \mathbf{z}_j | \mathbf{x}_t)} \left[ \log P(\{ y_t^{t+T} \}_{i=1}^N | \mathbf{u}_j, \mathbf{Z}_j) \right] + \sum_{j=1}^J \log P(\mathbf{u}_j | \mathbf{g}_j, \mathbf{z}_j) - \log q_j(\mathbf{u}_j | \mathbf{x}_t).
\]

The parameters of the inference distributions \( q_j \) and the components of the generative process are jointly learned via Stochastic Gradient Variational Bayes to minimize the ELBO loss [27]. We use the reparameterization trick for all sampling processes. The pseudo-code for training is available in Appendix. During inference, we sample from the joint distribution \( P(\{ y_t^{t+T} \}_{i=1}^N, \{ \mathbf{z}_j \}, \{ \mathbf{g}_j \}, \{ \mathbf{u}_j \}, \mathbf{U}, \mathbf{R}) \) to generate samples for the forecast distribution.

5 EXPERIMENTS

We evaluated our models on a workstation that runs on Intel Xeon 64 core processor with 128 GB memory on a Nvidia Tesla V100 GPU. Our model takes less than 6 GB of memory and takes 20-40 minutes of training time for each of the benchmarks. We have released the code and datasets publicly\(^1\) and describe in detail the hyperparameters in Appendix. Next, we describe the baselines, benchmark datasets and tasks as well as the evaluation metrics.

5.1 Setup

**Baselines.** We compare our model against the state-of-art probabilistic forecasting baselines along with some domain-specific baselines. The chosen forecasting baselines have shown state-of-art performance on a wide set of probabilistic forecasting tasks such as power consumption, air quality, traffic forecasting, health risk assessment, data center load estimation, etc. • SARIMA [21]: A classic time-series forecasting baseline based on ARIMA that accounts for seasonal shifts. • DeepAR [47]: A state-of-art, widely used RNN based probabilistic forecasting model that learns a parametric distribution. • Deep State Space Model (DSSM) [35]: A space state based model that uses neural networks to model transition and emission distribution of state space. • Deep Graph Factors (GraphDF) [6]: A deep probabilistic forecasting model that integrates relational information from graphs across time-series. In absence of explicit relational information, [6] proposes to use RBF kernel over the euclidean distance as edge weights over pairs of time-series, which we use as the GraphDF-RBF baseline. If explicit relational data in form of adjacency graph is available (in case of all benchmarks except power), we also evaluate using this adjacency graph as the GraphDF-Adj baseline. • Multi-modal Gaussian Process (MMGP): Similar to [5] we first pre-train a deterministic model combining deterministic embeddings from the encoders and aggregating embeddings with fixed learned weights to predict the output. Then we aggregate the embeddings as features to train a Gaussian Process to get probabilistic predictions.

For disease forecasting benchmarks, we also evaluate against state-of-art domain-specific forecasting models that use the same set of exogenous features as CAMul. We choose top performing deep learning models for flu-forecasting google-symptoms task: EpiDeep [1] and EpiFNP [24]. For the covid19 task, we also choose CMU-TS [9] and DeepCovid [45] as baselines; these leverage the CovDS dataset [25] and are top performing statistical models at the Covid-19 Forecast Hub organized by CDC [9]. We also evaluate against variants of CAMul, which enable us to examine the importance of Context-Specific View Selection and utility of our view-specific probabilistic modeling approach over the widely used method of probabilistic modeling on the fused deterministic embedding of view-specific representations (see Q2 of Section 5.2).

**Benchmarks.** We evaluate CAMul framework on time-series forecasting problem from a variety of domains involving diverse data views. For each benchmark, we describe the datasets used and the views corresponding to the features used from datasets.

1. **google-symptoms (Flu forecasting from Google symptoms):** We use aggregated and anonymized search counts of flu-related symptoms web searches by Google\(^2\) from each US state to predict incidence of influenza from 1 to 4 weeks ahead in future. Dataset: The flu incidence rate is represented by wILI (weighted Influenza related illness) values released weekly by CDC for 8 HHS regions of USA\(^3\). The aggregate symptom counts for over 400 symptons are anonymized and publicly released for each week since 2017 at county and state level by Google [17]. We choose to extract 14 symptoms related to influenza referred to in the CDC website\(^4\). We aggregate these counts for each HHS region and use it as exogenous features. We also use spatial adjacency data between HHS regions and time-related data (month and year) as features. For

\(^1\)https://github.com/AdityaLab/CAMul
\(^2\)https://predict.cdc.gov/post/5d8257befba2091084d47b4c
\(^3\)https://predict.cdc.gov/post/5db257b6ba2091084d47b4c
\(^4\)https://www.cdc.gov/flu/symptoms/symptoms.htm
forecasting wILI values for a given year, we use the training set from all past years for model training and hyperparameter tuning. Views: We use following diverse views for this benchmarks: 1. Defaulr Historical wILI view: This view contains reference points of symptoms features for all previous forecasting seasons for all HHS regions. We use the GRU encoder for this view. 2. HHS adjacency view: This is a graph feature view. We have 8 reference points corresponding to HHS regions. We aply use the GCN based latent encoder where the features \( F^{(j)} \) are one-hot encodings and the adjacency matrix \( A^{(j)} \) encodes the neighbourhood information between HHS regions that share a border. 3. Month view: This is a static feature view. We have 12 reference points for each month. We use an embedding layer to learn encodings for each reference point from one-hot embedding and use the feed-forward latent encoder. 2. covid19 (Covid-19 mortality forecasting): We evaluate our model on the challenging task of forecasting COVID-19 mortality for each of the 50 US states. Dataset: We use the mortality data and exogenous features of CovDS data used in [25, 45]. The exogenous features contain multiple kinds of data including hospital records, mobility, exposure and web-based survey data. We evaluate for duration of 7 months from June 2020 to December 2020 and again use only past weeks’ data to train before forecasting for 1 to 4 weeks ahead mortality. We follow the real-time forecasting setup of [45] and train the model separately for each week over all states using data available up to past week as training set. Views: 1. Default Mortality view: This is a sequence view where each reference point is a time-series of past mortality. 2. Line-list view: This is a multivariate sequence view that contains 7 weekly-collected features from traditional surveillance including hospitalization and testing. 3. Mobility and exposure view: Similarly we use the digitally collected signals measuring aggregate mobility and individuals collected from smartphones. For views 1,2 and 3 we use a GRU based encoder. 4. Demographic view: This is a static view where we encode each of the 50 states using 8 demographic features including average income, unemployment, population, average age and voting preferences. We use a feed-forward network for encoder. 5. State adjacency view: We construct a graph view of 50 states and encode spatial adjacency states. We use a GCN-based encoder. 3. power (Power consumption forecasting): We evaluate on power consumption data which is a standard forecasting benchmark. Dataset: The dataset [18] contains 260,640 measurements of power consumption of a household for a year using measurements from 3 meters with a total of 7 features. Our goal is to forecast the total active power consumption for 1 minute in future. We use the data from the first 9 months of measurement as training set and last 3 months as test set for forecasting. Views: 1. Default Past sequence view: We randomly sample single day sequence from each of the past months as reference sets. 2. Month view: Similar to Month view for google-symptoms benchmark. 3. Time of Day view: This is a static view. We divide the 24 hours of a day into 6 equal intervals and assign each of 6 reference point an interval. Similar to Month view we use an embedding layer on one-hod encodings to represent each reference point. 4. tweet (Tweet Topics prediction): The goal for this task is to evaluate the topic distribution of tweets in the future given topic distributions of past weeks similar to Shi et al. [48]. Dataset: We collect Covid-19 related tweets for 15 weeks that have a geographical tag to identify US state of the user. From the tweet text, we extract 30 topics using LDA [2] and allocate each tweet from a given week to each of the 30 topics it is most likely related for each state. Thus, we have a multivariate sequence dataset where for each of the 50 states, we have sequences containing topic distribution of tweets for each week. Similar to google-symptoms, for each forecasting for each year, we use data from past year as training set. We train a separate decoder module for each of the 30 topics and report the scores averaged over all topics. Views: 1. Default Past sequence view: Similar to other task the first view contains the past sequences for each state. 2. State adjacency view: We construct a state adjacency view similar to covid19 task. 3. Month view: Similar to Month view of power and google-symptoms. 4. Demographics view: Similar to Demographic view of covid19. Evaluation metrics: Given the large set of evaluation metrics proposed for both point-prediction and probabilistic predictions [20, 23, 51], we evaluate our model and baselines using carefully chosen metrics that are widely used in machine learning literature, relevant to our benchmarks and comprehensively evaluate both accuracy and calibration of models’ forecasts. Root Mean Squared Error (RMSE) is a popular metric used to evaluate accuracy of point prediction and is a better measure of robustness of model over metrics like MAE or MAPE since it is sensitive to instances of large errors. Interval Score (IS) is a standard score used in evaluation of accuracy of probabilistic forecasts in epidemiology [43]. IS measures the negative log likelihood of a fixed size interval around the ground truth under the predictive distribution:

\[
IS(\hat{p}_y, y) = -\int_{y-L}^{y+L} \log \hat{p}_y(\tilde{y}) d\tilde{y}.
\]

Confidence Score (CS) introduced in [24] measures the overall calibration of predictions of a model \( M \) similar to [30]. We first calculate the fraction \( k_m(c) \) of prediction distributions that cover the ground truth at each confidence interval \( c \). A perfectly calibrated model has \( k_m(c) \) very close to \( c \). Therefore, CS is defined as:

\[
CS(M) = \int_0^1 |k_m(c) - c| dc.
\]

which is approximated by summation over small intervals [24]. Cumulative Ranked Probability Score (CRPS) is a widely used standard metric for evaluation of probabilistic forecasts that generalizes mean average error to probabilistic forecasting [15]. Since it is a proper scoring rule that is minimized if the prediction distribution matches, on expectation, the true distribution, CRPS measures both accuracy and calibration. Given ground truth \( y \) and the predicted probability distribution \( \hat{p}(Y) \), let \( \hat{F}_y \) be the CDF. Then, CRPS is defined as:

\[
CRPS(\hat{F}_y, y) = \int_{-\infty}^{\infty} (\hat{F}_y(\tilde{y}) - 1(\tilde{y} > y))^2 d\tilde{y}.
\]

5.2 Results

Q1: Does CAMul provide well-calibrated accurate probabilistic forecasts across all benchmarks?
We evaluate our model and baselines on the four diverse benchmarks described in Section 5.1. We ran the experiments 20 times for each of the models for all tasks and reported the mean scores. Specifically, for covid19 and google-symptoms we performed a comprehensive evaluation across all regions in US for 1-4 weeks ahead forecasts and reported the average results. The results are summarized in Tables 1. CAMul models significantly outperform the baselines in both accuracy and calibration scores. Specifically for the hard-disease-tasking views we consistently see over 25% improvement in RMSE and CRPS score, and over 50% improvement in interval score over baselines. In case of power and tweet, we observe 28% and 24% improvement in CRPS scores over second-best model and 56% and 35% improvement in interval score. Performing significance test (t-test) with α = 1% shows that CAMul is significantly better than other models in all scores except those highlighted in bold in Table 1. Further, applying post-hoc calibration methods [30, 49] on baselines also does not affect the significance of our results (Table 4 in Supplementary).

**Q2:** Effect of multi-source stochastic modelling and context-sensitive dynamic view selection on CAMul’s performance

We evaluate the efficacy of 1) Context-Specific Dynamic Views Selection and 2) probabilistic modeling of each data view via Multi-view Latent Probabilistic Encoders and VSCG. We compare the attention based dynamic view selection of CAMul with two other variants a) **CAMul-C concatenates** VSCG-based latent embeddings from all views, b) **CAMul-S learns a static weight w(j)** for each view j and combines the VSCG-based embeddings: \( \hat{\eta}_i = \sum_j w(j) \eta^{(j)}_i \), c) To test the efficacy of stochastic modelling of latent embeddings, the variant **CAMul-D uses deterministic view-specific latent variables**: We use the latent encoders’ mean rather than sampling from Gaussian as Eqn. 1 and use a cross-attention layer over reference points instead of VSCG to derive the view specific latent variables as a weighted summation of reference points. The evaluation scores of the variants are shown in Table 1. We see that the original configuration of CAMul with dynamic view selection and multi-source probabilistic modeling outperforms the variants in all benchmarks. Using deterministic multi-source latent embeddings, in particular, drastically decreases the scores emphasizing the efficacy of uncertainty modeling.

**Q3:** Does CAMul handle information and uncertainty from multiple views to provide better performance?

Table 1: Evaluation scores (over 20 runs) for CAMul and baselines for all benchmarks. We performed t-test with α = 1%. Best scores are in bold and are statistically significantly better than other models. CAMul consistently performs the best with over 24% improvement in accuracy and over 35% improvement in calibration over the best baselines.

|                | RMSE | CRPS | CS   | IS    |
|----------------|------|------|------|-------|
| **covid19**    |      |      |      |       |
| SARIMA         | 91.3±3.7 | 69.1±4.2 | 0.14±0.015 | 6.57±0.074 |
| DeepAR         | 49.1±6.9 | 48.5±7.1 | 0.19±0.015 | 3.72±0.043 |
| DSSM           | 54.7±5.7 | 58.6±3.5 | 0.19±0.006 | 4.13±0.013 |
| RNP            | 52.9±5.3 | 71.8±5.6 | 0.32±0.015 | 5.31±0.042 |
| MMGP           | 48.7±4.2 | 43.2±2.4 | 0.23±0.015 | 4.27±0.022 |
| GraphDF-RBF    | 51.3±3.7 | 43.2±2.1 | 0.19±0.013 | 3.41±0.035 |
| GraphDF-Adj    | 48.5±4.2 | 49.1±3.4 | 0.23±0.021 | 3.79±0.223 |
| DeepCovid      | 38.3±5.2 | 39.8±6.6 | 0.22±0.015 | 3.63±0.046 |
| CAMul          | 27.3±3.5 | 25.8±4.1 | 0.14±0.011 | 3.08±0.015 |
| **google-symptoms** |      |      |      |       |
| SARIMA         | 1.72±0.021 | 1.62±0.016 | 0.46±0.005 | 3.04±0.032 |
| DeepAR         | 0.68±0.024 | 0.97±0.021 | 0.15±0.006 | 2.89±0.021 |
| DSSM           | 0.78±0.026 | 1.03±0.015 | 0.15±0.007 | 3.08±0.016 |
| RNP            | 0.82±0.043 | 0.95±0.035 | 0.41±0.004 | 3.87±0.018 |
| MMGP           | 1.24±0.077 | 1.02±0.031 | 0.26±0.003 | 2.82±0.042 |
| GraphDF-RBF    | 0.73±0.031 | 1.05±0.019 | 0.11±0.003 | 1.83±0.027 |
| GraphDF-Adj    | 0.91±0.027 | 1.10±0.042 | 0.12±0.005 | 2.64±0.053 |
| EpiDeep        | 0.98±0.053 | 1.07±0.062 | 0.29±0.006 | 5.55±0.084 |
| EpiRNP         | 0.64±0.048 | 0.52±0.057 | 0.05±0.004 | 0.67±0.12 |

Table 2: Comparison of CRPS score of CAMul with all views, the default view, two best views and the best baseline.
all benchmarks, the best view is always the default view. We see that model with all views is clearly the best performing. Moreover, using only the best view sometimes leads to lower performance compared to the best performing baseline which has access to data from all views. Therefore, we conclude that CAMul can handle multiple views that in turn lead to significantly better performance.

Q4: Do the weights of each view from View Selection module correspond to predictive utility of the view?

We provide specific case-studies to show the efficacy of the view selection module in selecting useful views specific to the input sequence by studying the attention weights. We observe that the default view 1 containing time-series sequences was almost always the most highly weighted views and was necessary for model to perform reasonably. Next, we study the correlation between attention weights of other views and the efficacy of the views. We assess the efficacy of a view by training variant of CAMul with two views: the default view and view we are interested in. We compare the CRPS scores of these variants to the attention weights (Figure 3) and see that the CRPS scores are inversely proportional to the view selection module weight. This further shows that the view selection module, indeed, selects the most informative views on average to improve the performance of our CAMul models.

Q5: Does the dynamic view-selection module adapt across time to select informative views?

We introduce CAMul, a general non-parametric generative framework for multi-source, multi-modal probabilistic forecasting. Our framework successfully tackles the challenge of modeling probabilistic information and uncertainty from diverse data views of multiple modalities across multiple domain benchmarks. CAMul outperformed both state-of-art general probabilistic baselines as well as top domain-specific baselines, resulting in over 25% improvement in accuracy and calibration metrics. Our case-studies also empirically showed the importance of joint probabilistic modeling and context-sensitive integration from multiple views. It automatically adapts to select more useful data views as seen in the covid19 and google-symptoms case studies. CAMul can easily be applied to any multi-source and multi-modal forecasting task by using appropriate encoders to encode a variety of data where capturing multi-source uncertainty is important. Analysis of view selection for specific tasks can also help understand data reliability, the disparity in data quality across sensitive parameters. As future work, our work can also be extended to tasks such as anomaly detection, change point detection and time-series segmentation where drastic variation in confidence intervals and view selection weights can be a useful predictive indicator of important behaviors.

Acknowledgments: This work was supported in part by the NSF (Expeditions CCF-1918770, CAREER IIS-2028586, RAPID IIS-2027862, Medium IIS-1955883, Medium IIS-2106961, CCF-2115126, Small III-2008334), CDC MinD program, ORNL, ONR MURI N00014-17-1-2656, faculty research awards from Facebook, Google and Amazon and funds/computing resources from Georgia Tech.

Figure 3: CRPS is inversely correlated with attention weights of View selection module.

The attention weights learned by the view selection module in Eqn. 5 informs the learned importance of views that are used for aggregating view-aware latent embeddings. We observe that the default view 1 containing time-series sequences was almost always the most highly weighted views and was necessary for model to perform reasonably. Next, we study the correlation between attention weights of other views and the efficacy of the views. We assess the efficacy of a view by training variant of CAMul with two views: the default view and view we are interested in. We compare the CRPS scores of these variants to the attention weights (Figure 3) and see that the CRPS scores are inversely proportional to the view selection module weight. This further shows that the view selection module, indeed, selects the most informative views on average to improve the performance of our CAMul models.

Q4: Do the weights of each view from View Selection module correspond to predictive utility of the view?

Figure 4: Sequence view is more important during peak weeks

An example is shown in Figure 4 for the 2018 flu season. We see that the peak week is at week 40. The weights corresponding the past sequence view sees an increase on weeks 38 - 42 whereas weights of other views decrease.

6 CONCLUSION
REFERENCES

[1] Bijaya Adhikari, Xinfeng Xu, Naren Ramakrishnan, and B Aditya Prakash. 2019. Epidep: Exploiting embeddings for epidemic forecasting. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 577–586.

[2] David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. the journal of machine Learning research 3 (2003), 993–1022.

[3] John Budd, Benjamin S Miller, Eric M Manning, Vasileios Lamos, Mengdie Zhang, Michael Beldestein, Geraint Rees, Vincent C Emery, Molly M Stevens, Neil Keegan, et al. 2020. Digital technologies in the public-health response to COVID-19. Nature medicine 26, 8 (2020), 1183–1192.

[4] Serina Chang, Emma Pierson, Pang Wei Koh, Jaline Gerardin, Beth Redbird, David Gruyke, and Jure Leskovec. 2021. Mobility network models of COVID-19 explain inequities and inform reopening. Nature 589, 7840 (2021), 82–87.

[5] Chacha Chen, Junjie Liang, Fengfeng Ma, Lucas Glass, Jimeng Sun, and Cao Xiao. 2021. UNITE: Uncertainty-based Health Risk Prediction Leveraging Multi-sourced Data. In Proceedings of the Web Conference 2021. 217–226.

[6] Hongjie Chen, Ryan A Rossi, Kanak Mahadik, Sungchul Kim, and Hoda Eldardiry. 2021. Categorical reparameterization vs late fusion in multimodal convolutional neural networks. In Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 35. 1449–1457.

[7] Georges Hebrail and Alice Berard. 2012. Individual household electric power processing systems 28 (2015). 2980–2988.

[8] Estee Y Cramer, Velma K Lopez, Jarad Niemi, Glover E George, Jeffrey C Cegan, Ian D Dettwiller, William P England, Matthew W Farthing, Robert H Hunter, Brandon Lafferty, et al. 2021. Evaluation of individual and ensemble probabilistic forecasts of COVID-19 mortality in the US. medRxiv (2021).

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018).

[10] Vijay Ekambaram, Kushagra Manglik, Sumanta Mukherjee, Surya Shravan Kumar, Georges Hebrail, and Alice Berard. 2020. Early vs late fusion in multimodal convolutional neural networks. IEEE transactions on knowledge and data engineering 33, 1 (2021), 103, 9 (2015), 1099–1108.

[11] Hojjat Salehinejad, Sharan Sankar, Joseph Barfett, Errol Colak, and Shahrokh Valaee. 2017. Recent advances in recurrent neural networks. arXiv preprint arXiv:1807.06222 (2018).

[12] Konrad Gadzicki, Razieh Khamsehashari, and Christoph Zetzsche. 2020. State-sharing sparse hidden markov models for personalized sequences. IEEE transactions on knowledge and data engineering 32, 12 (2020), 2357–2366.

[13] Ramnath Kumar, Shweta Yadav, Raminta Danilutsyte, Francois Lamy, Krishnaprasad Thurirayan, Usha Lokala, and Amit Seth. 2020. edarkinf: Unsupervised multi-view learning for sbih account detection. In Proceedings of The Web Conference 2020. 1955–1965.

[14] Bijaya Adhikari, Xinfeng Xu, Naren Ramakrishnan, and B Aditya Prakash. 2019. Variational Gaussian Processes for Regression. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 3110–3118.

[15] Alexander Rodríguez, Jiaming Cui, Jiajia Xie, Javen Ho, Pulak Chakraborty,等多项 Neuro-symbolic branching programs in parallel. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 125–151.

[16] Syama Sundar Rangapuram, Matthias W Seeger, Jan Gasthaus, Lorenzo Stella, Ian D Dettwiller, William P England, Matthew W Farthing, Robert H Hunter, Brandon Lafferty, et al. 2021. Evaluation of individual and ensemble probabilistic forecasts of COVID-19 mortality in the US. medRxiv (2021).

[17] Alexander Rodríguez, Nikhil Muralidhar, Bijaya Adhikari, Anika Tabassum, and Yoshua Bengio. 2015. A recurrent latent variable model for sequential data. The journal of machine Learning research 16 (2015), 1–29.

[18] Nature medicine 26, 8 (2019), 3146–3154.

[19] Estee Y Cramer, Velma K Lopez, Jarad Niemi, Glover E George, Jeffrey C Cegan, Ian D Dettwiller, William P England, Matthew W Farthing, Robert H Hunter, Brandon Lafferty, et al. 2021. Evaluation of individual and ensemble probabilistic forecasts of COVID-19 mortality in the US. medRxiv (2021).

[20] Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. 2020. Mining of massive data sets. Cambridge university press.

[21] Alexander Rodríguez, Anika Tabassum, Jiaming Cui, Jiajia Xie, Javen Ho, Pulak Chakraborty,等多项 Neuro-symbolic branching programs in parallel. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 125–151.

[22] Georges Hebrail and Alice Berard. 2012. Individual household electric power processing systems 28 (2015). 2980–2988.

[23] Estee Y Cramer, Velma K Lopez, Jarad Niemi, Glover E George, Jeffrey C Cegan, Ian D Dettwiller, William P England, Matthew W Farthing, Robert H Hunter, Brandon Lafferty, et al. 2021. Evaluation of individual and ensemble probabilistic forecasts of COVID-19 mortality in the US. medRxiv (2021).

[24] Alexander Rodríguez, Nikhil Muralidhar, Bijaya Adhikari, Anika Tabassum, and Yoshua Bengio. 2015. A recurrent latent variable model for sequential data. The journal of machine Learning research 16 (2015), 1–29.

[25] Syama Sundar Rangapuram, Matthias W Seeger, Jan Gasthaus, Lorenzo Stella, Ian D Dettwiller, William P England, Matthew W Farthing, Robert H Hunter, Brandon Lafferty, et al. 2021. Evaluation of individual and ensemble probabilistic forecasts of COVID-19 mortality in the US. medRxiv (2021).

[26] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014).

[27] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114 (2013).

[28] Thomas Kipf and M. Welling. 2016. Semi-Supervised Classification with Graph Convolutional Networks. ICLR (2016).

[29] Georges Hebrail and Alice Berard. 2012. Individual household electric power processing systems 28 (2015). 2980–2988.

[30] Estee Y Cramer, Velma K Lopez, Jarad Niemi, Glover E George, Jeffrey C Cegan, Ian D Dettwiller, William P England, Matthew W Farthing, Robert H Hunter, Brandon Lafferty, et al. 2021. Evaluation of individual and ensemble probabilistic forecasts of COVID-19 mortality in the US. medRxiv (2021).

[31] Alexander Rodríguez, Nikhil Muralidhar, Bijaya Adhikari, Anika Tabassum, and Yoshua Bengio. 2015. A recurrent latent variable model for sequential data. The journal of machine Learning research 16 (2015), 1–29.
[50] Patara Trirat and Jae-Gil Lee. 2021. DF-TAR: A Deep Fusion Network for Citywide Traffic Accident Risk Prediction with Dangerous Driving Behavior. In Proceedings of the Web Conference 2021. 1146–1156.

[51] Alexander Tsyplakov. 2013. Evaluation of probabilistic forecasts: proper scoring rules and moments. Available at SSRN 2236605 (2013).

[52] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In Advances in neural information processing systems. 5998–6008.

[53] Zilong Wang, Zhaohong Wan, and Xiaojun Wan. 2020. Transmodality: An end2end fusion method with transformer for multimodal sentiment analysis. In Proceedings of The Web Conference 2020. 2514–2520.

[54] Xiaoqiang Yan, Shizhe Hu, Yuqiao Mao, Yangdong Ye, and Hui Yu. 2021. Deep multi-view learning methods: a review. Neurocomputing (2021).

[55] Qiwei Zhong, Yang Liu, Xiang Ao, Binbin Hu, Jinghua Feng, Jiayu Tang, and Qing He. 2020. Financial defaulter detection on online credit payment via multi-view attributed heterogeneous information network. In Proceedings of The Web Conference 2020. 785–795.

[56] Lingxue Zhu and Nikolay Laptev. 2017. Deep and confident prediction for time series at uber. In 2017 IEEE International Conference on Data Mining Workshops (ICDMW). IEEE, 103–110.
Supplementary for the paper "CAMul: Calibrated and Accurate Multi-view Time-Series Forecasting"

We have released the code for CAMul models and anonymized datasets at: https://github.com/AdityaLab/CAMul.

AEXAMPLES OF MULTI-VIEW PROBABILISTIC LATENT ENCODERS

We provide examples of Probabilistic Latent Encoders for data views of different modalities that are used in the paper. Due to generality of our framework, variety of neural encoders can be used specific to data view and type of task.

Static features: We encode the static features of fixed dimensions of form $\mathbf{R}_k^{(j)} \in \mathbb{R}^d$. We employ a feed-forward network to capture the parameters of distribution.

Sequences: To capture the latent embedding of sequence $\mathbf{R}_k^{(j)} = \{R[0]_k^{(j)}, R[1]_k^{(j)}, \ldots, R[T]_k^{(j)}\}$, such as in the default view, we leverage neural sequence encoders GRU [7] as:

$$\{\hat{z}[t]_k^{(j)}\}_{t=0}^T = \text{GRU}(\{R[t]_k^{(j)}\}_{t=0}^T)$$

where $\{\hat{z}[t]_k^{(j)}\}_{t=0}^T$ are the intermediate states of GRU.

In order to capture long-term relations and prevent over-\emph{fit}ting on last terms of sequence we employ self-attention layer introduced in [52]:

$$\{\alpha[0], \alpha[1], \ldots, \alpha[T]\}_d = \text{Self-Atten}((\hat{z}[t]_k^{(j)})_{t=0}^T)$$

$$\mu_k^{(j)}(z_i), \sigma_k^{(j)}(z_i) = g_f^j(\sum_{t=0}^T \alpha[t]_d z[t]_d^{(j)})$$

where $g_f^j$ is a single feed-forward layer.

Graph data: If the data type of view $j$ contain relations, i.e., the reference sets have an inherent graph structure denoted by $(\mathbf{A}^{(j)}, \mathbf{F}^{(j)})$, where $\mathbf{A}^{(j)} \in \mathbb{R}^{N \times N}$ is the adjacency matrix (potentially with weights) and $\mathbf{F}^{(j)} = \{f^{(j)}_{i=1} \}_{i=1}^{N}$ are feature vectors of fixed dimensions, then we can use a Graph Neural network architecture [28] to encode the relations in $\mathbf{A}^{(j)}$:

$$\{\hat{z}_i^{(j)}\}_{i=1}^{N} = \text{GCN}(\mathbf{A}^{(j)}, \{f^{(j)}_{i=1} \}_{i=1}^{N})$$

Then, we use feed forward layer to derive the distribution parameters:

$$\mu_i^{(j)}, \sigma_i^{(j)} = g_f^j(\hat{z}_i^{(j)})$$

B IMPLEMENTATION DETAILS

We used numpy for data processing and PyTorch for model training and inference. We use the properscoring library [33] to implement the CRPS evaluation.

B.1 Data pre-processing

Scaling values: Since time-series and exogenous features can have wide range of values, we normalize the values of each features with mean 0 and variance 1. We derive the scaling factors for training dataset and apply the transformation to test set during inference.

Chunking time-series for training: The long training sequences are split using the shingling technique [34, 35] where we fix a window size $W = 10$ and randomly sample chunks of $W$ size from the full sequence over the interval $[1, t – W – 1]$ and record the $\tau$ ahead value as ground truth. Note that the reference set of default view 1 still contains the full length sequence of training set, only we use the split sequence as input for training set.

B.2 Hyperparameters

We describe the hyperparameters for CAMul for all benchmarks. Multi-view probabilistic encoders

For the feed-forward networks of static view, we used a 3 layer network with 60 hidden units. We also used 60 hidden units for GRU of sequence views and used a bi-directional version of GRU. For graph views, we used a 2-layer GCN network with 60 hidden units. The final layer outputs 120 dimensional vector which we split into mean and variance. We apply exponentiation on the variance vector to make it positive. The latent embeddings $z_i^{(j)}$ for all views thus have dimension of 60.

View specific correlation graph The networks $l_1^p$ and $l_2^p$ are a layer feed forward network with 60 hidden units. Thus, the dimension of view-aware latent embedding is 60.

Dynamic view-selection module We pass the view-aware latent embeddings and $z^{(1)}_i$ each through a single 60-hidden unit feed forward layer $(h_1, h_2)$ before computing the cross attention weights. The combined-view embedding $\bar{u}_i$ is a 60 dimensional vector.

Forecast decoder We use a 3 layer feed forward layer that inputs the latent embedding of sequence $z^{(1)}_i$, $\bar{u}_i$ and has 60 hidden units in each hidden layer with final layer outputting 2 scalars $\mu(y_i), \sigma(y_i)$.

Throughout all layers of CAMul we apply exponentiation to variance vector/scalar to make it positive. We also use ReLU activation for hidden layers unless specified otherwise. We also use ADAM optimizer [26] for parameter updates.

Note on hyperparameter selection. We sampled a validation set containing 10% of randomly selected chunks of sequences from training set. This was used for model hyperparameter tuning. We describe splitting of dataset into training and test set in Section 5.1 below for each benchmark.

We found that the performance was not significantly sensitive to model architecture, batch size or learning rate. We searched over the space of $[30, 60, 120, 250]$ for hidden units and mostly optimized for faster convergence. We also searched over $\{10, 20, 50, 80\}$ for batch-size and found $20, 20, 10, 50$ to be most optimal for tweet, covid19, google-symptoms, power respectively. We used 0.005 as learning rate. We also used early stopping with patience of 150 epochs for faster training. For power and google-symptoms we observed that CAMul took less than 700 epochs to converge whereas it took 2000 and 1000 epochs for covid19 and tweet. Regarding seed selection, we initialized random seeds to 0 to 19 for numpy and pytorch for the 20 trials done for each benchmarks and didn’t observe any significant variation in scores.
Algorithm 2: Training Algorithm for CAMul

Input : Training sequences \( \{Y_N\}_{i=1}^{N} = \{X_i^{(1)}\}_{i=1}^{N} \), target labels \( \{y[t + r]_{i=1}^{N}\} \) for each view \( j \in \{2, \ldots, K\} \), reference points \( \{R_j^{(j)}\}_{i=1}^{N} \) for each view \( j \)

1. for \( i \in \{1, \ldots, N\} \) do
   2. for \( j \in \{1, \ldots, K\} \) do
      3. /* Stochastic embeddings from latent stochastic encoder */
         4. Sample \( z_i^{(j)} \) as Eqn 1 using encoder \( g_0 \);
         5. for \( k \in R_j^{(j)} \) do
            6. Sample \( z_k^{(j)} \) using encoder \( g_0 \);
      7. end /* Sample edges for SVCG */
      8. for \( k \in R_j^{(j)} \) do
         9. Add \( (k, i) \) to \( G^{(j)} \) with probability \( k(z_k^{(j)} ; z_i^{(j)}) \);
      10. end Derive \( \mu(u_i^{(j)}), \sigma(u_i^{(j)}) \) from sampled edges \( \{k : (k, i) \in G^{(j)}\} \) as Eqn 4;
      11. /* Sample from Variational distribution */
         12. Sample \( q_i^{(j)} \) from variational distribution \( q_j \);
   13. end /* Aggregate all view-aware embeddings using Dynamic view-selection module */
   14. Compute importance weights \( \{a_i^{(j)}\}_{j=1}^{K} \) using cross-attention as Eqn 5 from \( \{u_i^{(j)}\}_{j=1}^{K} \);
   15. Compute the combined view embedding \( \tilde{u}_i^{(j)} = \sum_{j=1}^{K} a_i^{(j)} \tilde{u}_i^{(j)} \);
   16. /* Final output distribution */
      17. Derive \( \mu(y_i), \sigma(y_i) \) from \( z_i^{(j)} \) and \( \tilde{u}_i^{(j)} \) via the decoder (Eqn 7);
      18. /* Sample ELBO Loss */
         19. Compute \( L_1 = \log P(y[t + r]_{i=1}^{N}|\mu(y_i), \sigma(y_i)) \);
         20. Compute \( L_2 = \sum_{j=1}^{K} \log P(\tilde{u}_i^{(j)}|\mu(u_i^{(j)}), \sigma(u_i^{(j)})) - \log q_i^{(j)}(\tilde{u}_i^{(j)}|X_i^{(j)})) \);
         21. Accumulate gradient for loss \( L = -(L_1 + L_2) \);
   22. end
   23. end
24. Periodically update the weights of all modules of CAMul from accumulated gradients using ADAM;

Table 3: Avg. Attention weight of View selection module during June and July for some populous US states.

Table 4: Evaluation scores of CAMul and baselines after applying post-hoc calibration. We use Isotonic regression [30] and Distribution calibration [49] methods. The evaluation scores of CAMul is statistically significantly better and does not change much due to post-hoc methods, implying that our approach produces well-calibrated forecasts without need for such post-hoc correction methods.