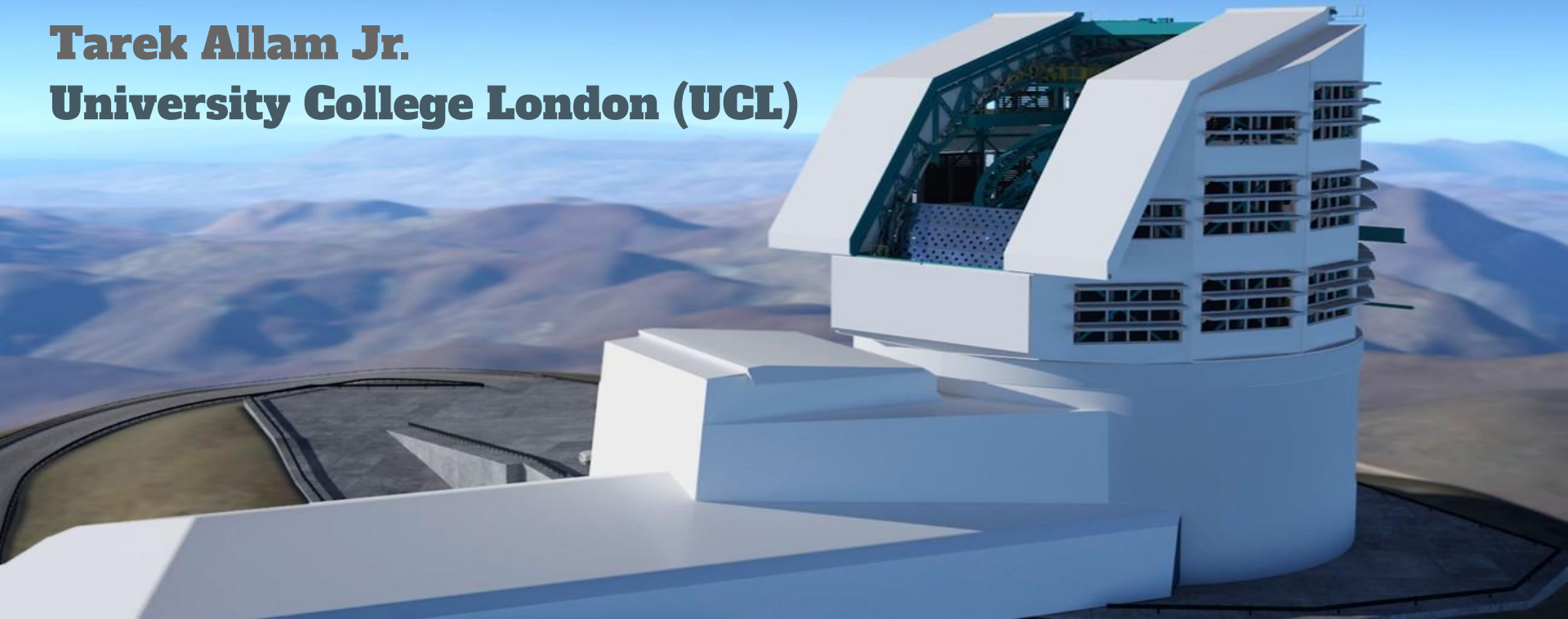
Low-latency High-throughput Classification using Deep Model Compression

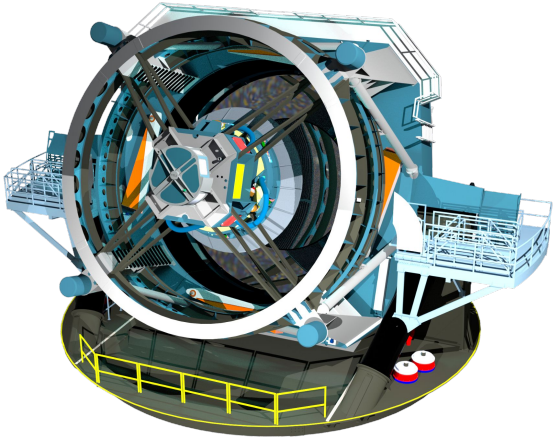
Tarek Allam Jr.
University College London (UCL)

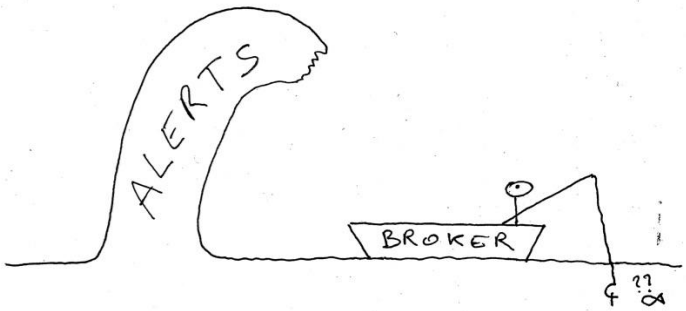


Motivation: A Deluge of Data

Overview

- 10 million alerts, per night!
- Machine Learning methods are now critical
- Accurate and fast classification required for follow up
- Desire for fast re-training of models





Peloton et al. 2020

Deep Learning to the Rescue!?

The death of feature engineering?

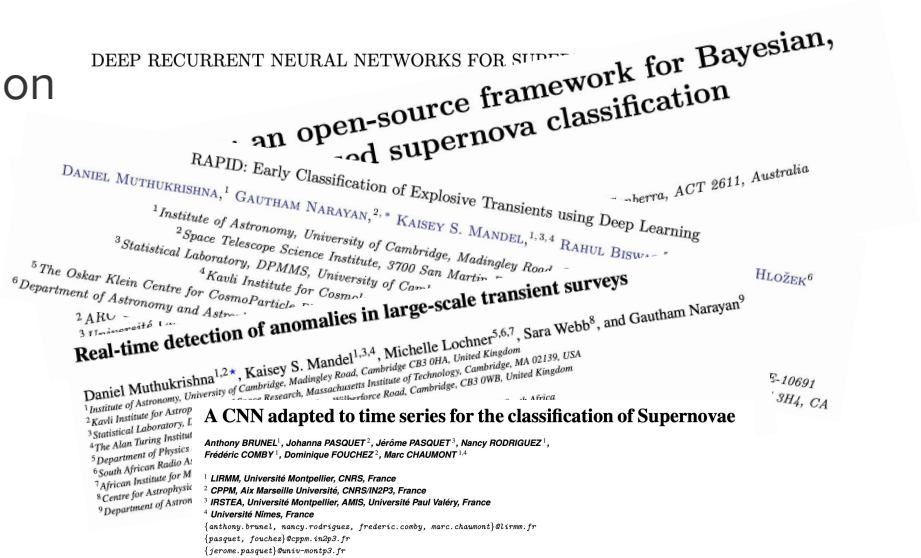
- Exploiting inherent time-series information

RNNs (inc. SuperNNova, RAPID)

CNNs (inc. TCN)

Transformers (inc. t2)

• • •


DEEP RECURRENT NEURAL NETWORKS FOR SUP

an open-source framework for Bayesian,
od supernova classification

RAPID: Early Classification of Explosive Transients using Deep Learning

DANIEL MUTHUKRISHNA, [1] GAUTHAM NARAYAN, [2,*] KAISEY S. MANDEL, [1,3,4] RAHUL BISW

[1] Institute of Astronomy, University of Cambridge, Madingley Road, Cambridge CB3 0HA, United Kingdom
[2] Space Telescope Science Institute, 3700 San Martin
[3] Statistical Laboratory, DPMMS, University of Cam
[4] Kavli Institute for Cosmo
[5] The Oskar Klein Centre for CosmoParticle
[6] Department of Astronomy and Astro

Real-time detection of anomalies in large-scale transient surveys

HLOŽEK[6]

Daniel Muthukrishna[1,2,*], Kaisey S. Mandel[1,3,4], Michelle Lochner[5,6,7], Sara Webb[8], and Gautham Narayan[9]

[1] Institute of Astronomy, University of Cambridge, Madingley Road, Cambridge CB3 0HA, United Kingdom
[2] Kavli Institute for Astrop
[3] Statistical Laboratory, D
[4] The Alan Turing Institut
[5] Department of Physics
[6] South African Radio As
[7] African Institute for M
[8] Centre for Astrophysic
[9] Department of Astron

Research, Massachusetts Institute of Technology, Cambridge, MA 02139, USA
Cambridge, CB3 0WB, United Kingdom
Africa

5-10691
3H4, CA

A CNN adapted to time series for the classification of Supernovae

Anthony BRUNEL[1], Johanna PASQUET[2], Jérôme PASQUET[3], Nancy RODRIGUEZ[1], Frédéric COMBY[1], Dominique FOUCHEZ[2], Marc CHAUMONT[1,4]

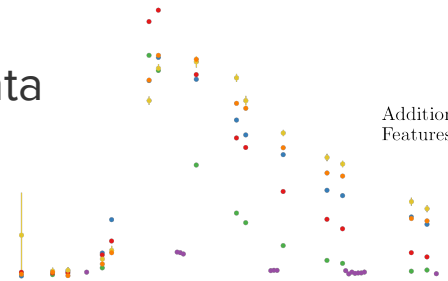
[1] LIRMM, Université Montpellier, CNRS, France
[2] CPPM, Aix Marseille Université, CNRS/IN2P3, France
[3] IRSTEA, Université Montpellier, AMIS, Université Paul Valéry, France
[4] Université Nîmes, France

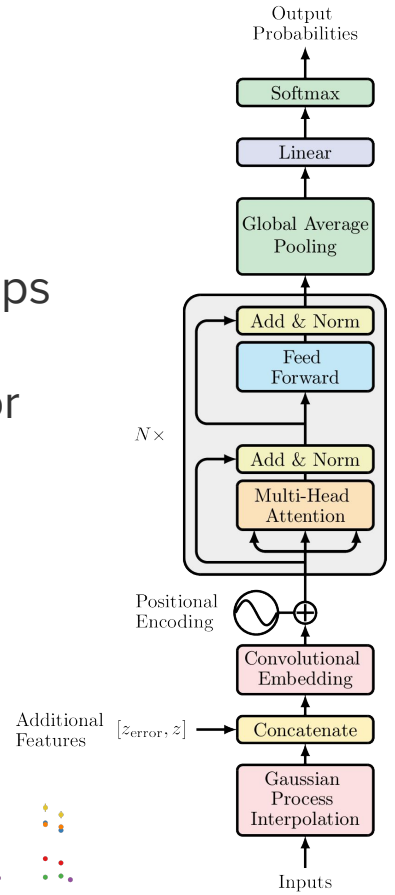
{anthony.brunel, nancy.rodrigues, frederic.comby, marc.chaumont}@lirmm.fr
{pasquet, fouchez}@cppm.in2p3.fr
{jerome.pasquet}@univ-montp3.fr

The Time-Series Transformer [t2]

Encoder++

- *Global Average Pooling* to allow for Class Activation Maps
- *Convolutional Embedding* maps time-series into a vector space
- *Concatenate* additional features
- *GP Interpolation* to handle irregular data

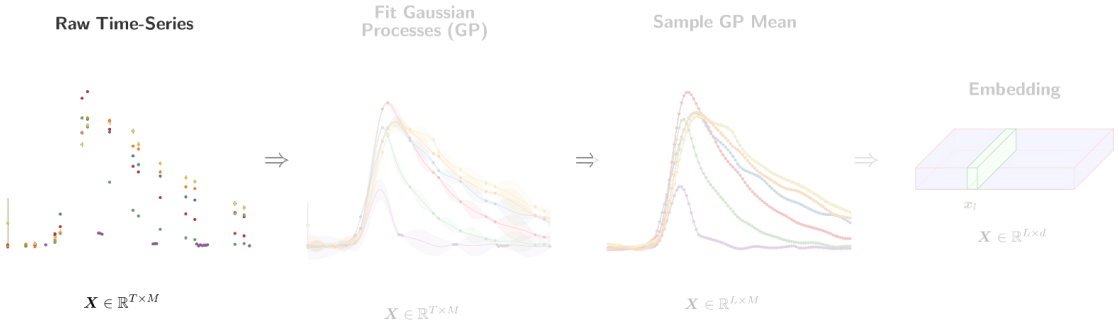


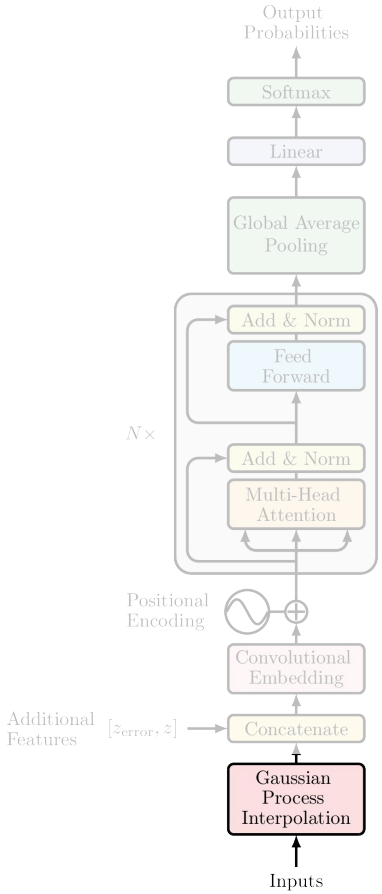


ML Pipeline

Gaussian Process Interpolation

- 2D-Matern kernel for Gaussian Process interpolation
- Evaluate at regular period, 100 points in our case

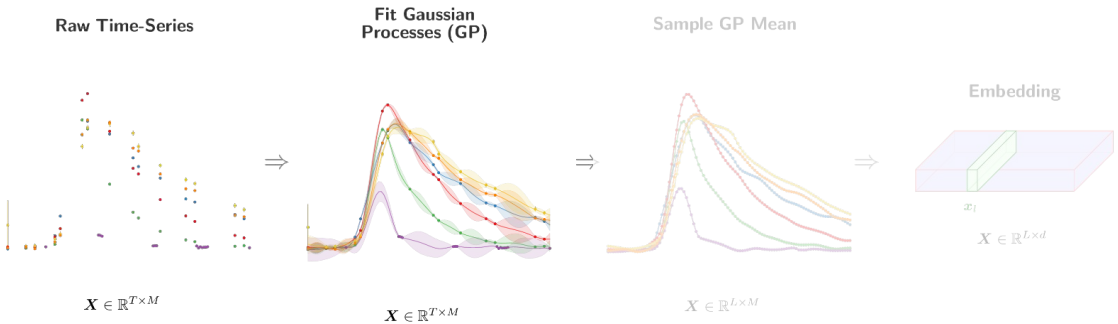


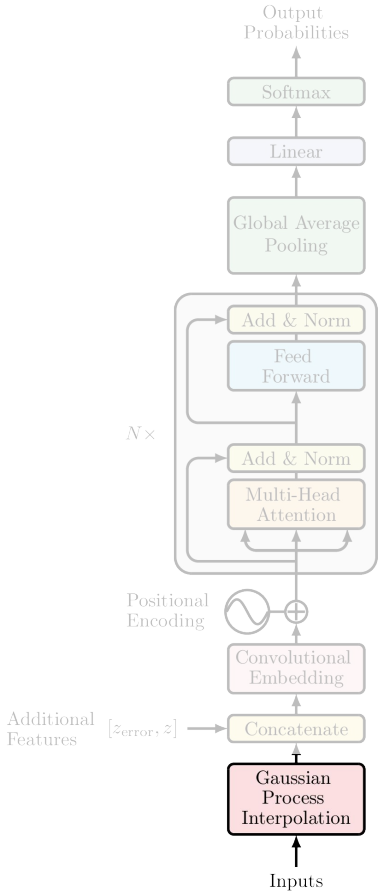


ML Pipeline

Gaussian Process Interpolation

- 2D-Matern kernel for Gaussian Process interpolation
- Evaluate at regular period, 100 points in our case

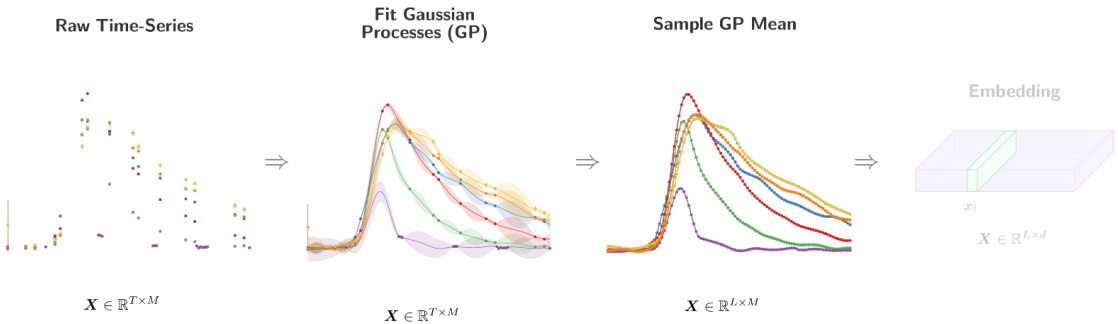


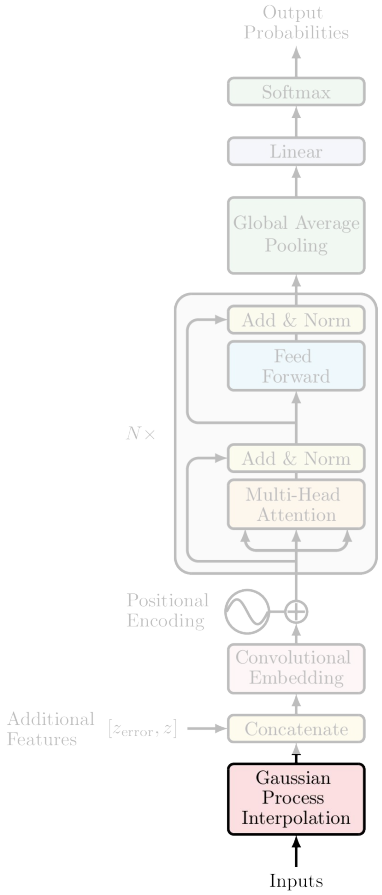


ML Pipeline

Gaussian Process Interpolation

- 2D-Matern kernel for Gaussian Process interpolation
- Evaluate at regular period, 100 points in our case

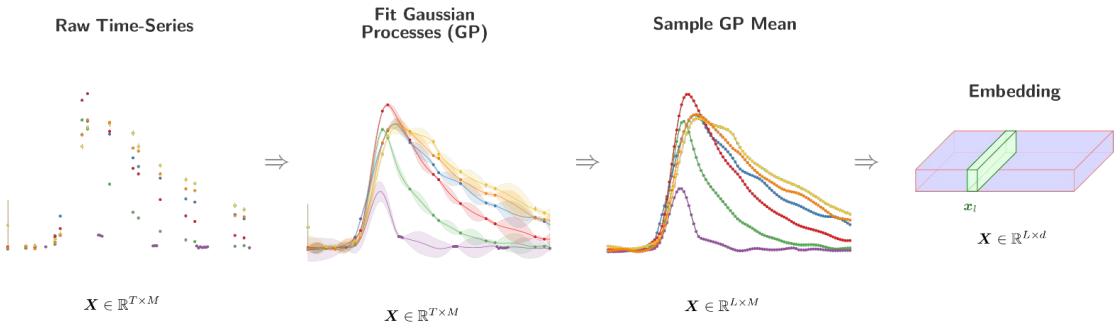


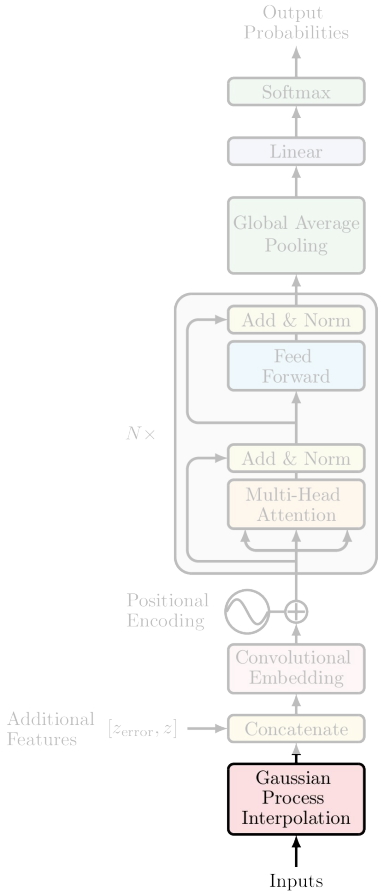


ML Pipeline

Gaussian Process Interpolation

- 2D-Matern kernel for Gaussian Process interpolation
- Evaluate at regular period, 100 points in our case



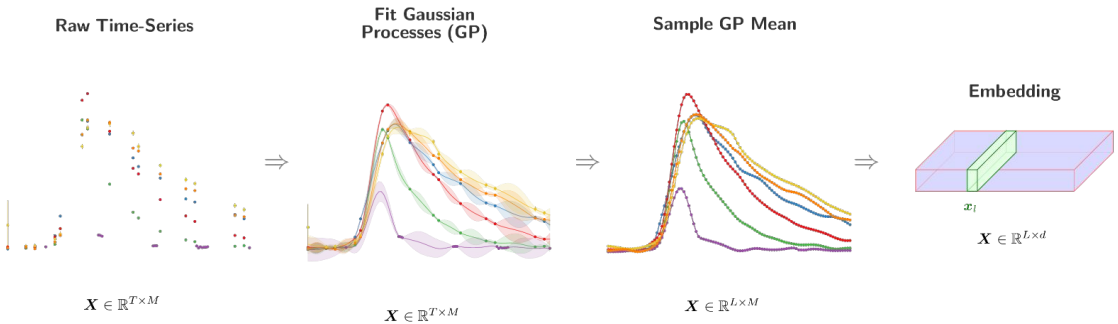


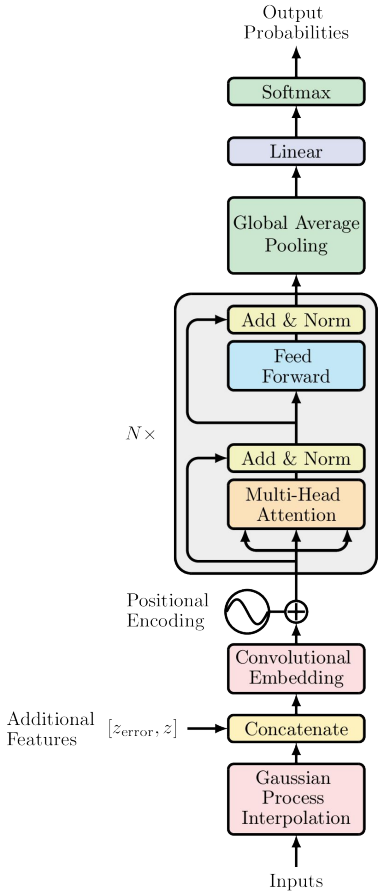
ML Pipeline

Gaussian Process Interpolation

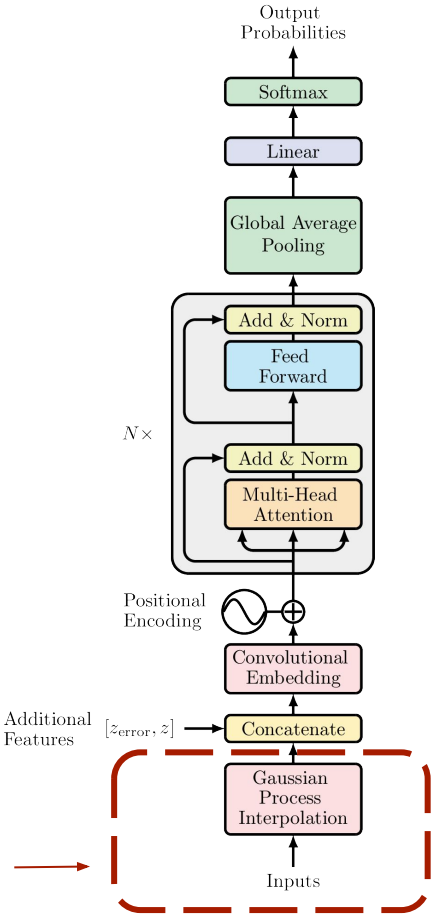
- 2D-Matern kernel for Gaussian Process interpolation
- Evaluate at regular period, 100 points in our case

Deep Learning Magic ...





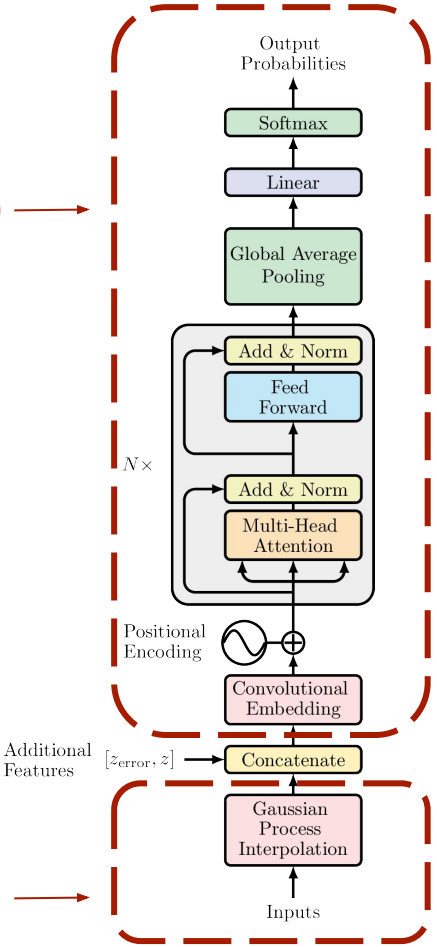
Understanding Bottlenecks



`fit_gps(raw_alert)` →

Understanding Bottlenecks

`model.predict(processed_alert)` →



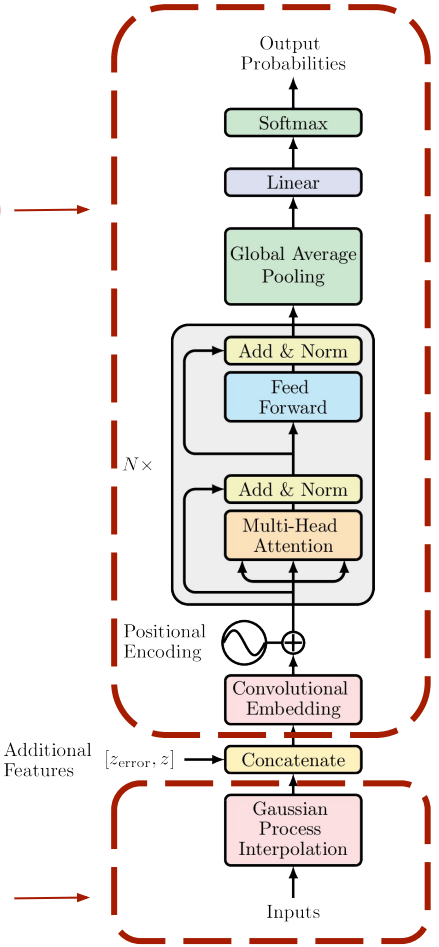
`fit_gps(raw_alert)` →

Understanding Bottlenecks

```
Total time: 5.85664 s
File: get_models.py
Function: get_model at line 29

Total time: 1.47076 s
File: ztf-load-run-lpa.py
Function: t2_probs at line 55

Line #    Hits       Time Per Hit \% Time Line Contents
==============================================================
...
206      16      139.6    8.7    9.5    df_gp_mean = generate_gp_all_objects()
...
...
...
...
...
212       8      180.8   22.6   12.3    X = df_gp_mean[cols]
213       8       12.3    1.5    0.8    X = rs(X)
...
...
...
217       8     1101.7  137.7   74.9    y_preds = model.predict(X)
```

model.predict(processed_alert) →

fit_gps(raw_alert) →

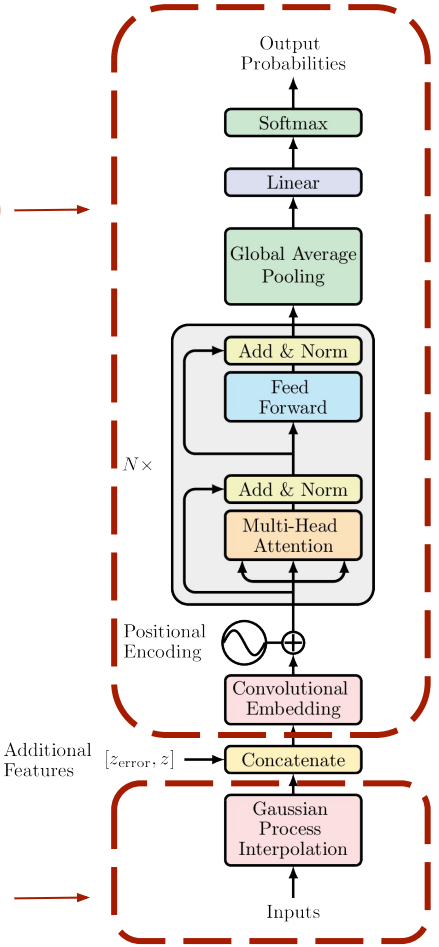


Understanding Bottlenecks

```
Total time: 5.85664 s
File: get_models.py
Function: get_model at line 29

Total time: 1.47076 s
File: ztf-load-run-lpa.py
Function: t2_probs at line 55

Line #    Hits       Time Per Hit \% Time Line Contents
==============================================================
...
206      16      139.6    8.7    9.5    df_gp_mean = generate_gp_all_objects()
...
...
...
...
...
212       8      180.8   22.6   12.3    X = df_gp_mean[cols]
213       8       12.3    1.5    0.8    X = rs(X)
...
...
...
217       8     1101.7  137.7   74.9    y_preds = model.predict(X)
```

model.predict(processed_alert) →

fit_gps(raw_alert) →

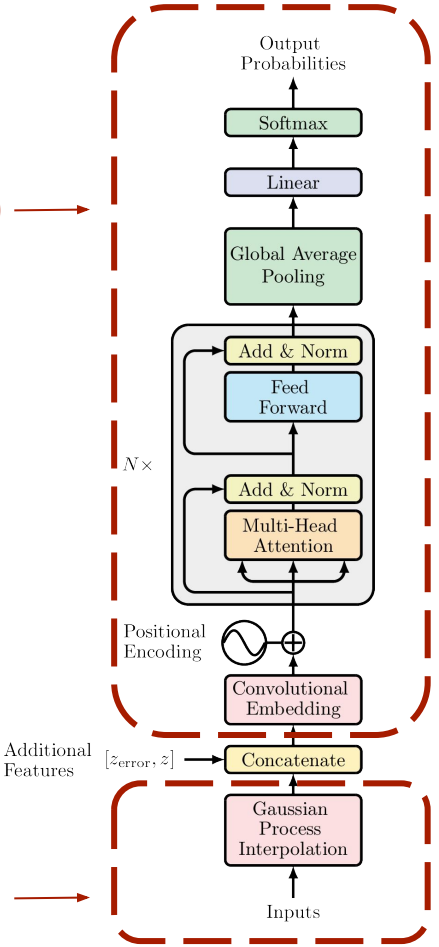


Understanding Bottlenecks

```
Total time: 5.85664 s
File: get_models.py
Function: get_model at line 29

Total time: 1.47076 s
File: ztf-load-run-lpa.py
Function: t2_probs at line 55

Line #    Hits       Time Per Hit \% Time Line Contents
==============================================================
...
206      16      139.6    8.7    9.5    df_gp_mean = generate_gp_all_objects()
...
...
...
...
...
212       8      180.8   22.6   12.3    X = df_gp_mean[cols]
213       8       12.3    1.5    0.8    X = rs(X)
...
...
...
217       8     1101.7  137.7   74.9    y_preds = model.predict(X)
```

model.predict(processed_alert) →

fit_gps(raw_alert) →



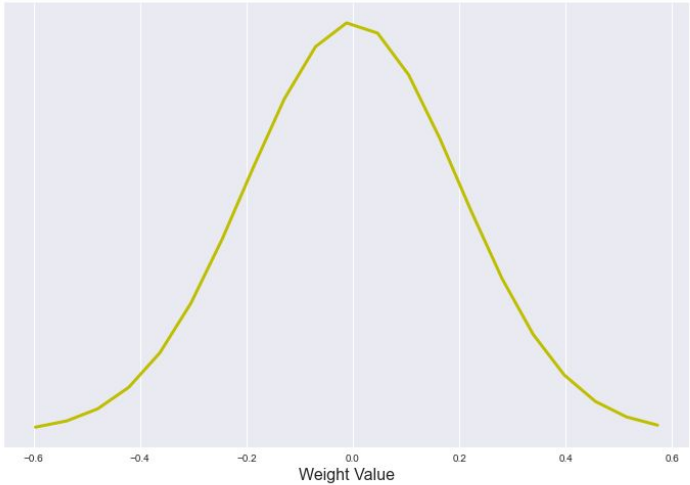
Deep Model Compression: An *Almost* Free Lunch

- Standout paper: Han et al. 2015: *Deep Compression*
  - Clustering
  - Pruning
  - Quantization
- Lossy process – expect some loss in accuracy (but improved compute and storage costs!)

Deep Model Compression: Clustering

Weight Clustering

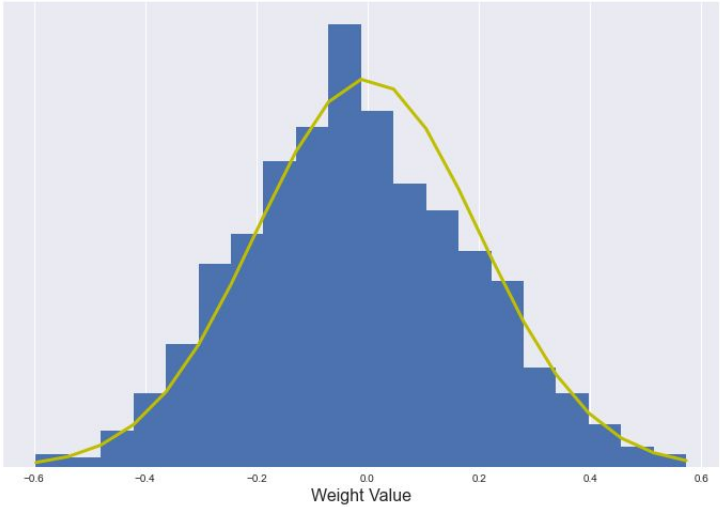
- Reduce number of unique weights with shared weight values
- Can then leverage Huffman encoding
- Note: best to avoid clustering early layers!



Deep Model Compression: Clustering

Weight Clustering

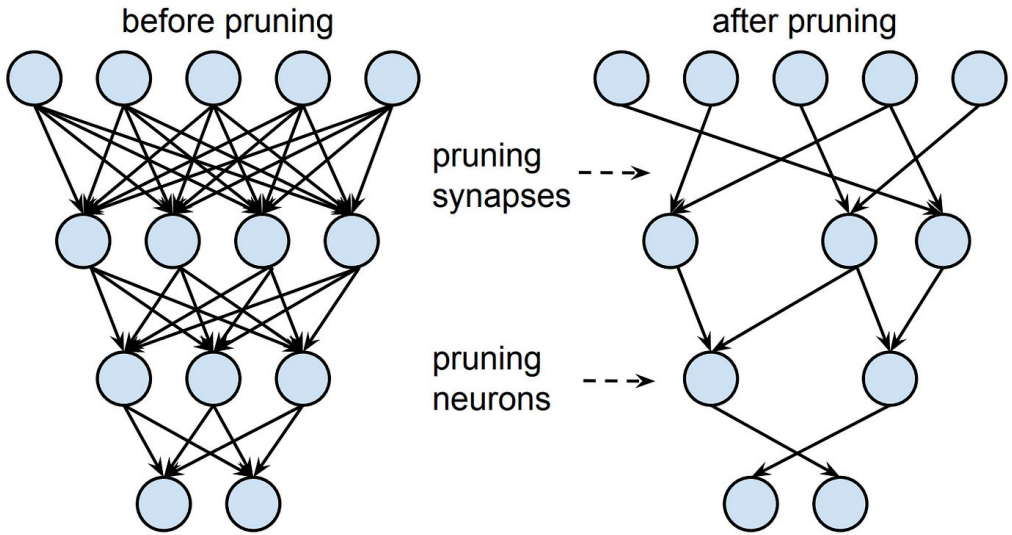
- Reduce number of unique weights with shared weight values
- Can then leverage Huffman encoding
- Note: best to avoid clustering early layers!



Deep Model Compression: Pruning

Weight Pruning

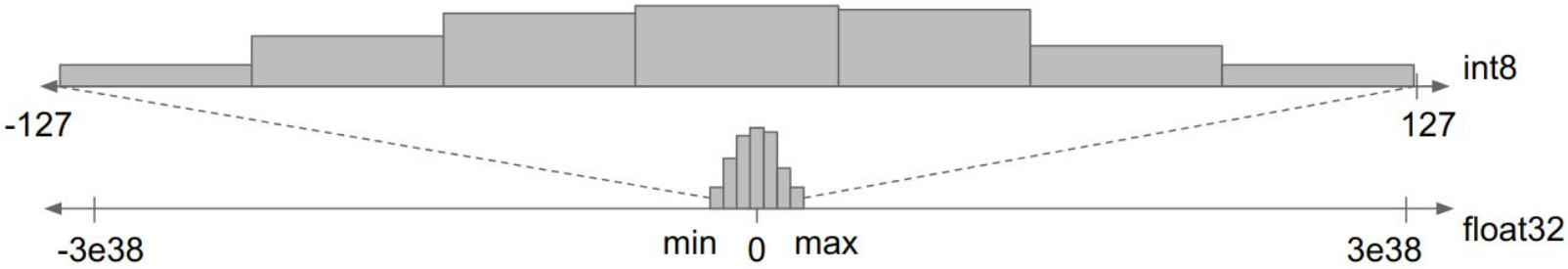
- Set low magnitude weights to zero
- Can then leverage Huffman encoding (again)
- Done *during* training



Deep Model Compression: Quantization

Weight Quantization

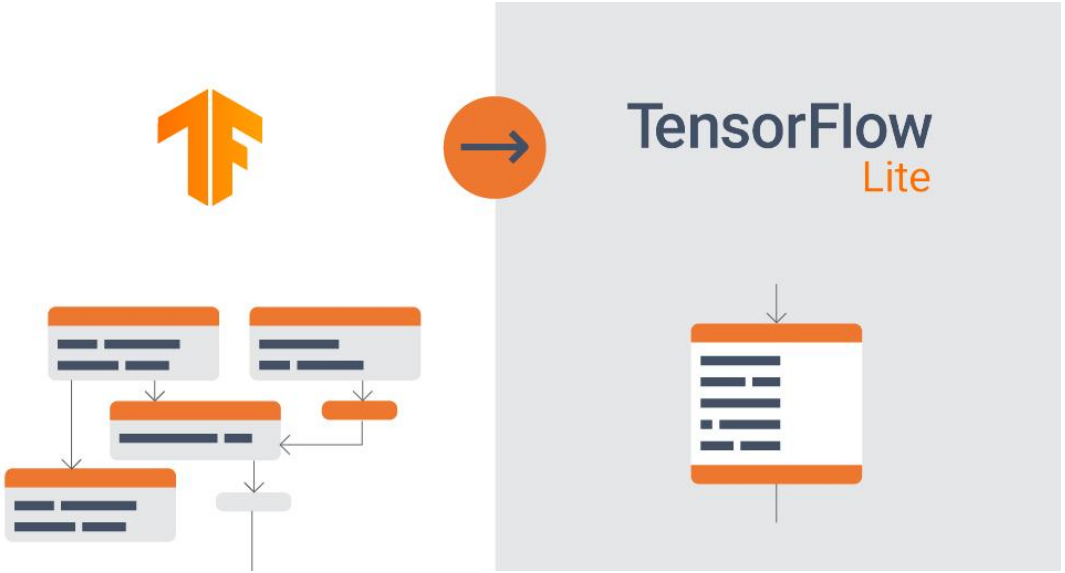
- Reduce precision of stored weight values
- Computations are still done at `float32`

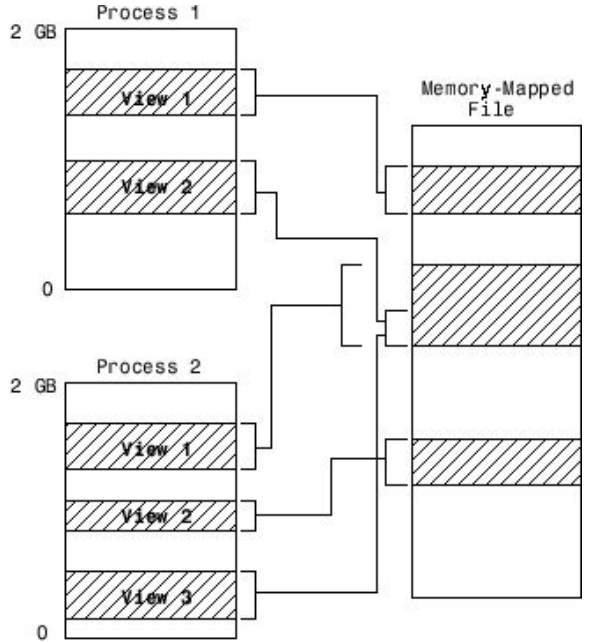


Efficient Frameworks & File Formats

From TensorFlow to TensorFlow-Lite

- Operator Fusion
- `ProtocolBuffers → FlatBuffers`





Local Tests

Compression Method	Model Size (kb)	Load Latency (s^{-3})	Inference Latency (s)	Loss
Baseline	1100	6324.145	0.333	0.968

Local Tests

Compression Method	Model Size (kb)	Load Latency (s^{-3})	Inference Latency (s)	Loss
Baseline	1100	6324.145	0.333	0.968
Baseline + Huffman	244	6015.565	0.224	0.968

Local Tests

Compression Method	Model Size (kB)	Load Latency (s^{-3})	Inference Latency (s)	Loss
Baseline	1100	6324.145	0.333	0.968
Baseline + Huffman	244	6015.565	0.224	0.968
Clustering	892	5559.868	0.227	0.836

Local Tests

Compression Method	Model Size (kB)	Load Latency (s^{-3})	Inference Latency (s)	Loss
Baseline	1100	6324.145	0.333	0.968
Baseline + Huffman	244	6015.565	0.224	0.968
Clustering	892	5559.868	0.227	0.836
Clustering + Pruning	688	5721.021	0.230	1.017

Local Tests

Compression Method	Model Size (kB)	Load Latency (s^{-3})	Inference Latency (s)	Loss
Baseline	1100	6324.145	0.333	0.968
Baseline + Huffman	244	6015.565	0.224	0.968
Clustering	892	5559.868	0.227	0.836
Clustering + Pruning	688	5721.021	0.230	1.017
Clustering + Huffman	240	4991.857	0.223	0.836

Local Tests

Compression Method	Model Size (kb)	Load Latency (s^{-3})	Inference Latency (s)	Loss
Baseline	1100	6324.145	0.333	0.968
Baseline + Huffman	244	6015.565	0.224	0.968
Clustering	892	5559.868	0.227	0.836
Clustering + Pruning	688	5721.021	0.230	1.017
Clustering + Huffman	240	4991.857	0.223	0.836
Clustering + Pruning + Huffman	128	5251.288	0.228	1.017

Local Tests

Compression Method	Model Size (kb)	Load Latency (s^{-3})	Inference Latency (s)	Loss
Baseline	1100	6324.145	0.333	0.968
Baseline + Huffman	244	6015.565	0.224	0.968
Clustering	892	5559.868	0.227	0.836
Clustering + Pruning	688	5721.021	0.230	1.017
Clustering + Huffman	240	4991.857	0.223	0.836
Clustering + Pruning + Huffman	128	5251.288	0.228	1.017
†Clustering	92	0.426	0.046	0.836

Local Tests

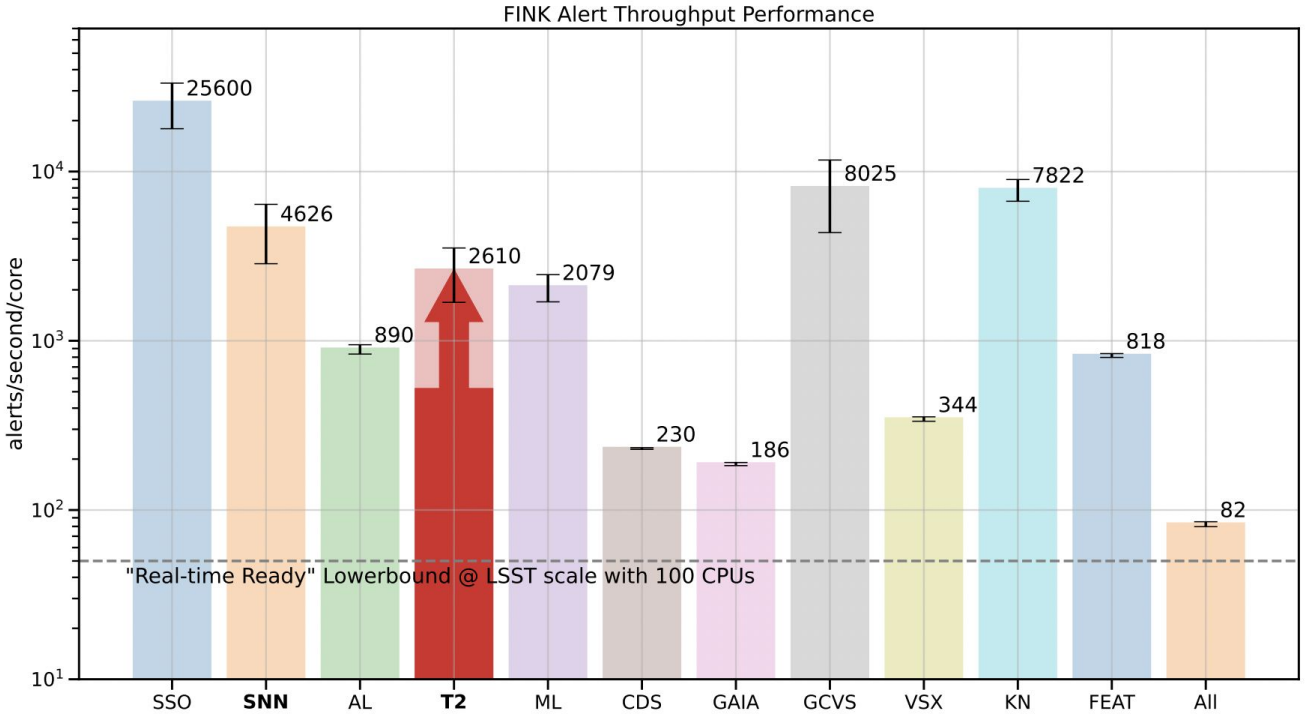
Compression Method	Model Size (kb)	Load Latency (s^{-3})	Inference Latency (s)	Loss
Baseline	1100	6324.145	0.333	0.968
Baseline + Huffman	244	6015.565	0.224	0.968
Clustering	892	5559.868	0.227	0.836
Clustering + Pruning	688	5721.021	0.230	1.017
Clustering + Huffman	240	4991.857	0.223	0.836
Clustering + Pruning + Huffman	128	5251.288	0.228	1.017
†Clustering	92	0.426	0.046	0.836
†Clustering + Quantization	**60**	**0.271**	**0.043**	**0.834**

Production Results

- One full night of ZTF alerts ~ 200K
- Require at least 2 points on light curve
- Averaged over 20 processing runs

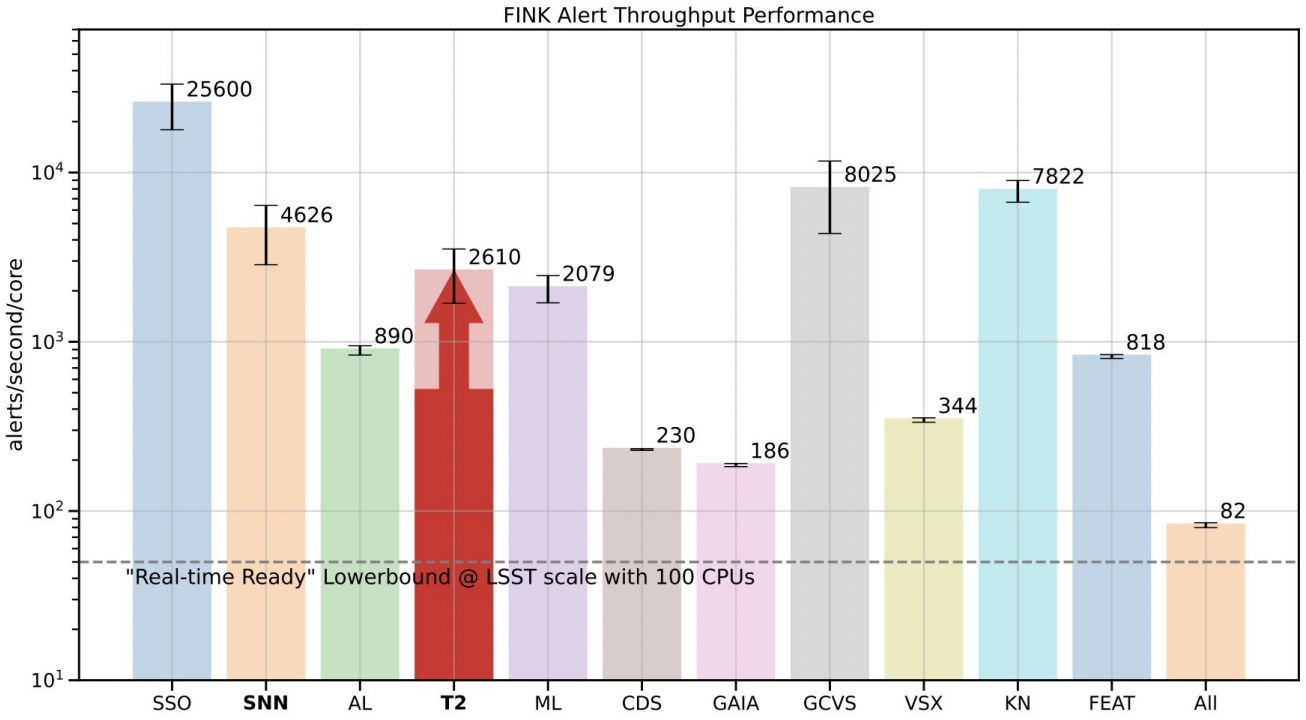
Production Results

- One full night of ZTF alerts ~ 200K
- Require at least 2 points on light curve
- Averaged over 20 processing runs
- **5x** throughput



Takeaways...

-


FINK Alert Throughput Performance
alerts/second/core
10^4
10^3
10^2
10^1
25600
4626
890
2610
2079
230
186
8025
344
7822
818
82
"Real-time Ready" Lowerbound @ LSST scale with 100 CPUs
SSO
SNN
AL
T2
ML
CDS
GAIA
GCVS
VSX
KN
FEAT
All

Questions?

Time for T2 in FINK

Considerations

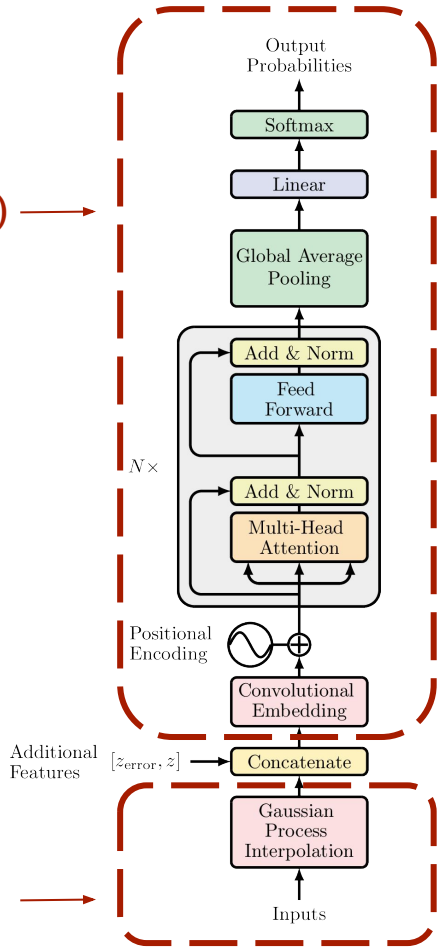
`model.predict(processed_alert)` →

- ZTF Alert stream, only *g* & *r* passbands vs LSST *ugrizy*
- Will only be raw-time series at this point (i.e. no-Z)

Approaching Implementation

- Stage One: Process Alerts
- Stage Two: Load TF Model for Inference

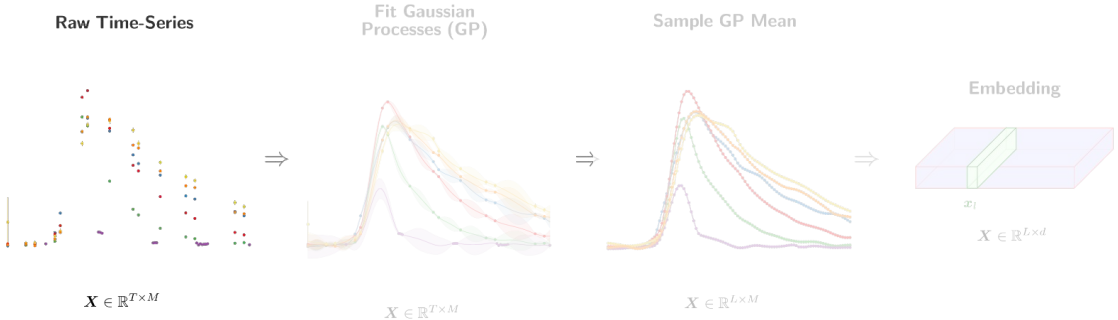
`fit_gps(raw_alert)` →

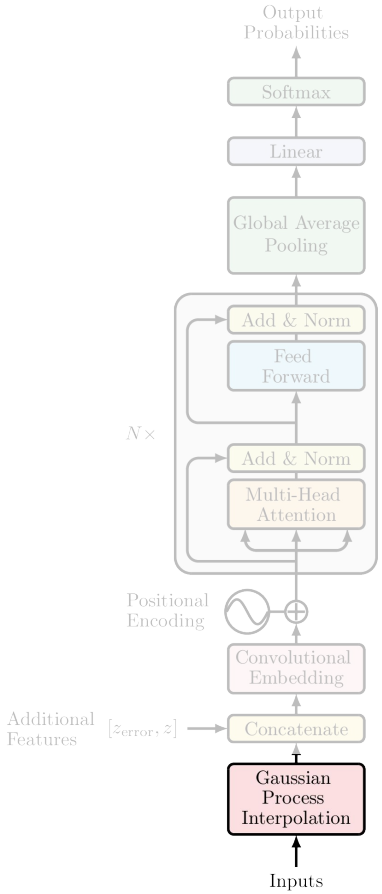


Dealing with Irregularly Sampled Multivariate Time-Series Data

Gaussian Process Interpolation

- 2D-Matern kernel for Gaussian Process interpolation
- Evaluate at regular period, 100 points in our case

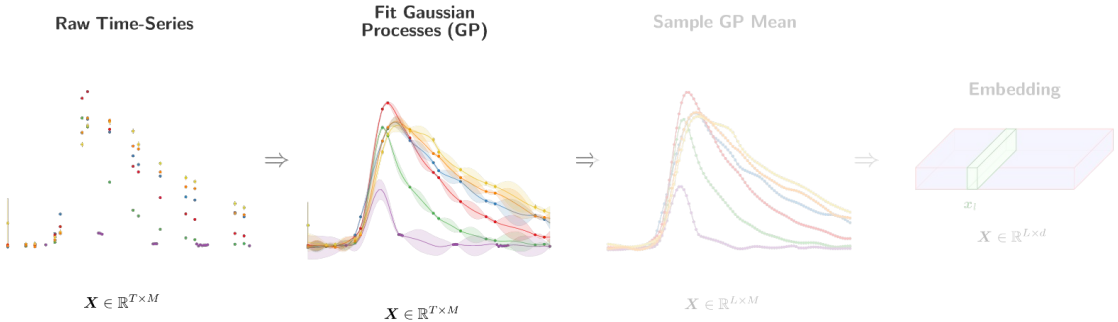


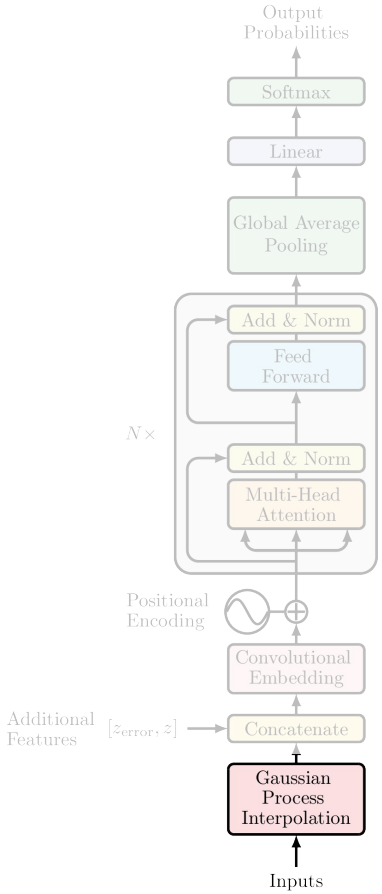


Dealing with Irregularly Sampled Multivariate Time-Series Data

Gaussian Process Interpolation

- 2D-Matern kernel for Gaussian Process interpolation
- Evaluate at regular period, 100 points in our case

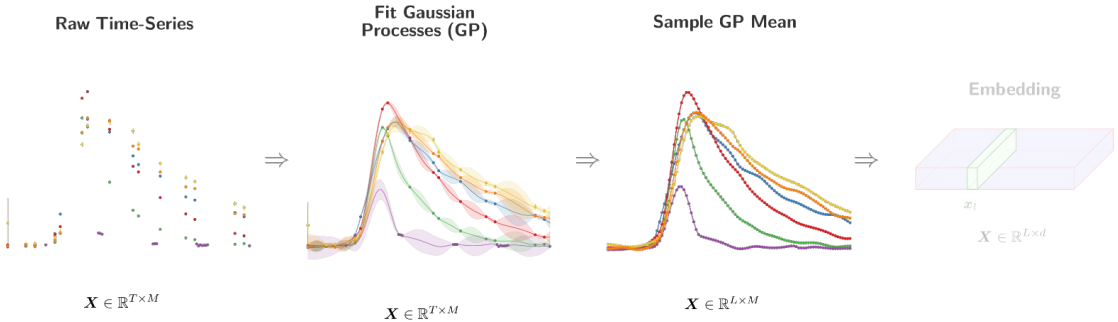


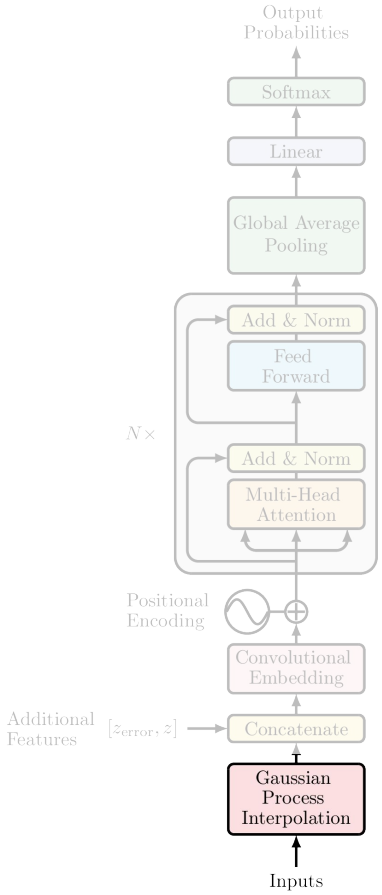


Dealing with Irregularly Sampled Multivariate Time-Series Data

Gaussian Process Interpolation

- 2D-Matern kernel for Gaussian Process interpolation
- Evaluate at regular period, 100 points in our case

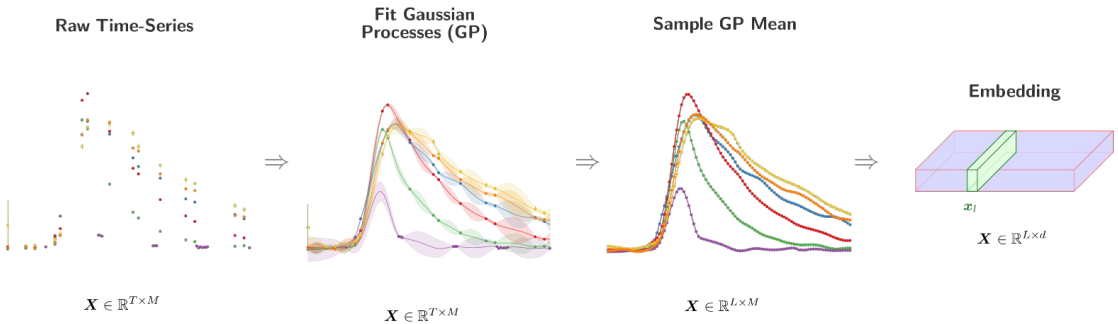


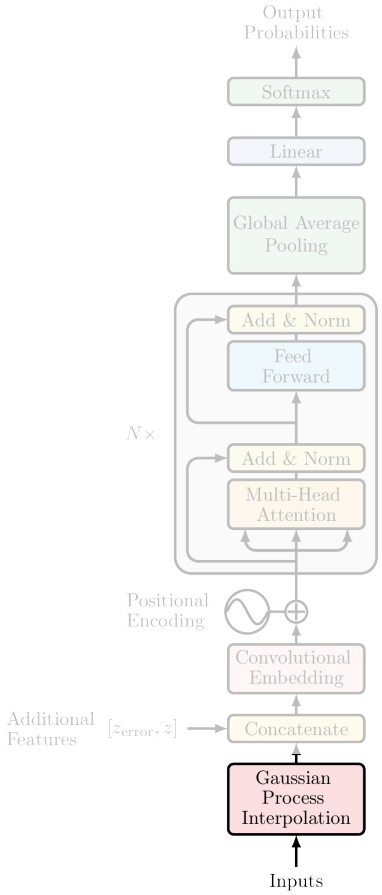


Dealing with Irregularly Sampled Multivariate Time-Series Data

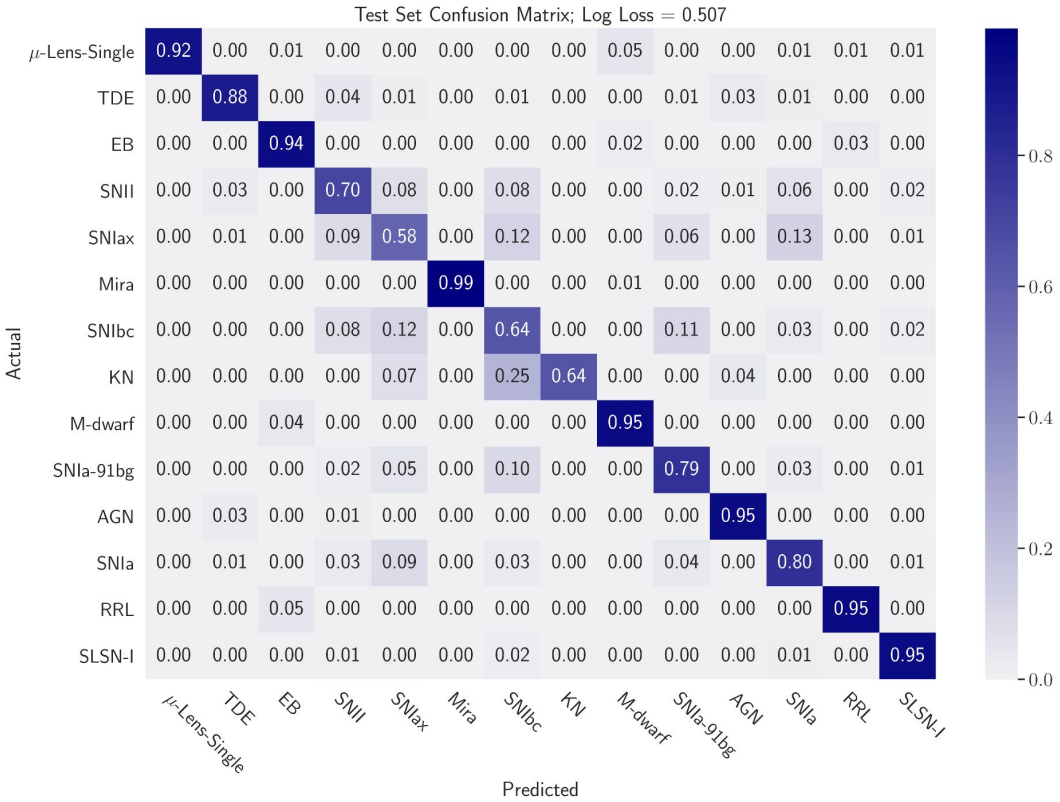
Gaussian Process Interpolation

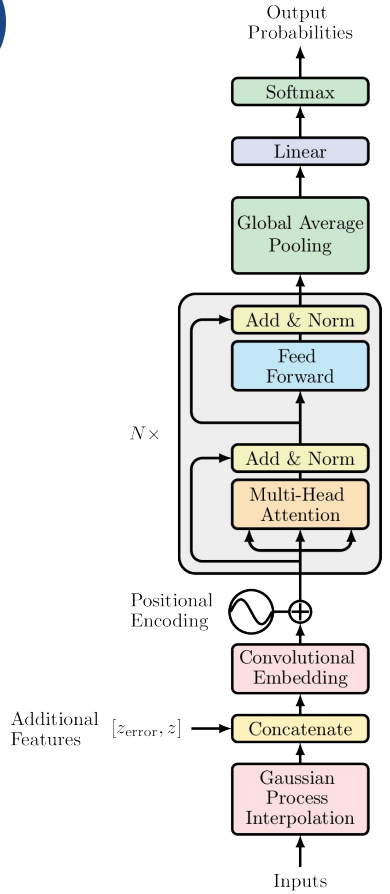
- 2D-Matern kernel for Gaussian Process interpolation
- Evaluate at regular period, 100 points in our case



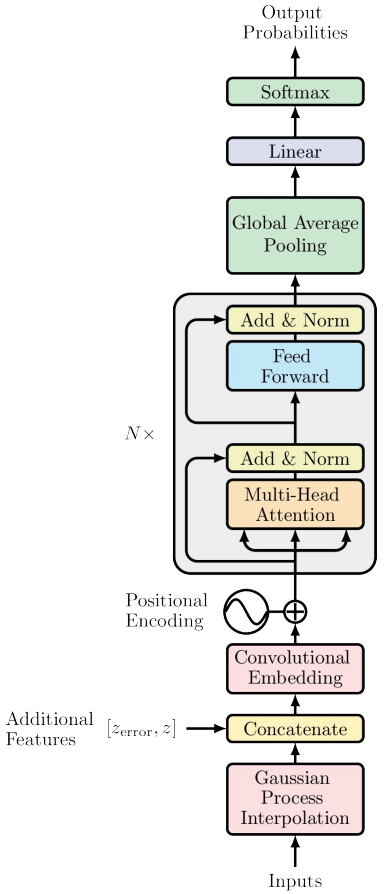


Performance and Results (*ugrizy+z*)





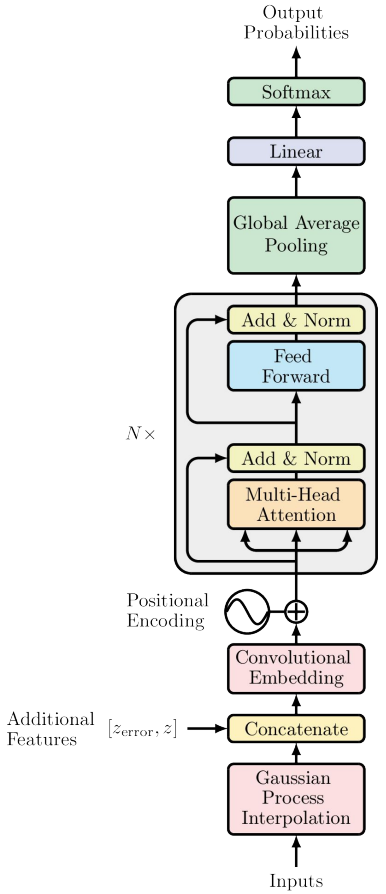
Time for T2 in FINK



Time for T2 in FINK

Considerations

- ZTF Alert stream, only *g* & *r* passbands vs LSST *ugrizy*
- Will only be raw-time series at this point (i.e. no-Z)



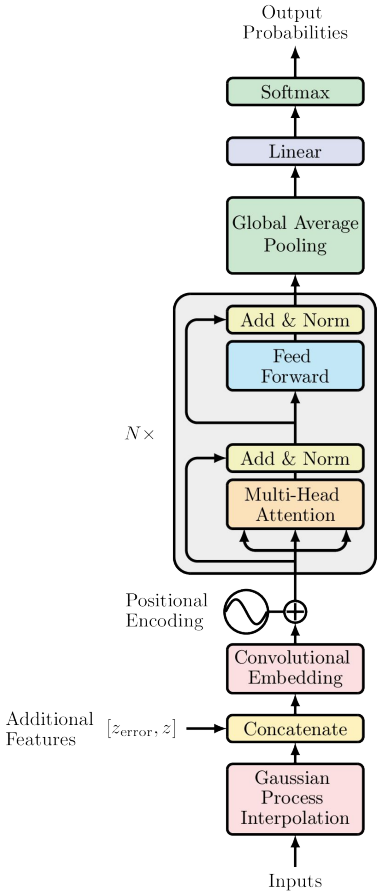
Time for T2 in FINK

Considerations

- ZTF Alert stream, only *g* & *r* passbands vs LSST *ugrizy*
- Will only be raw-time series at this point (i.e. no-Z)

Approaching Implementation

- Stage One: Process Alerts
- Stage Two: Load TF Model for Inference



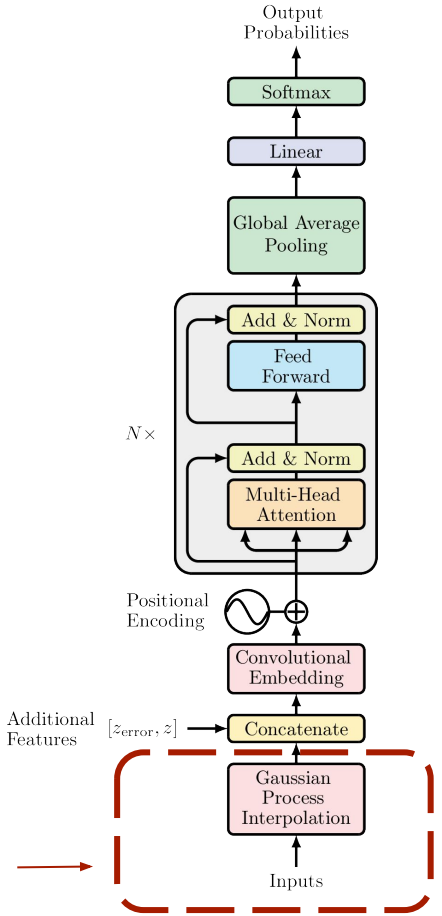
Time for T2 in FINK

Considerations

- ZTF Alert stream, only *g* & *r* passbands vs LSST *ugrizy*
- Will only be raw-time series at this point (i.e. no-Z)

Approaching Implementation

- Stage One: Process Alerts
- Stage Two: Load TF Model for Inference



`fit_gps(raw_alert)` →

Time for T2 in FINK

Considerations

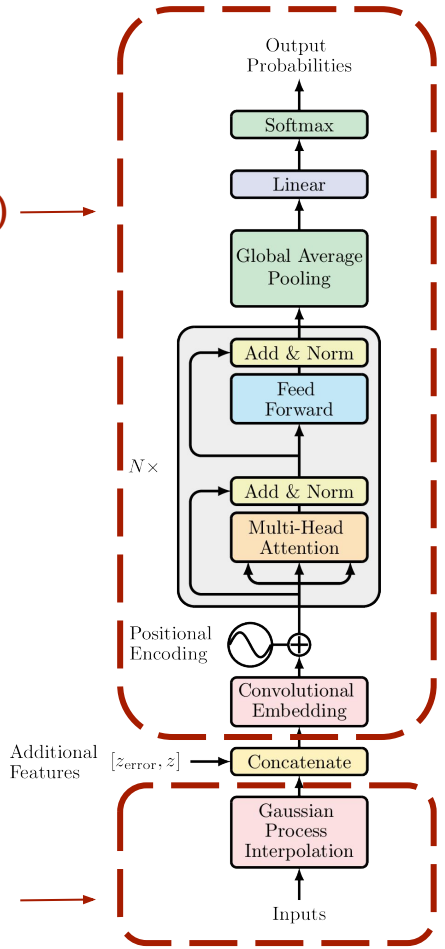
`model.predict(processed_alert)` →

- ZTF Alert stream, only *g* & *r* passbands vs LSST *ugrizy*
- Will only be raw-time series at this point (i.e. no-Z)

Approaching Implementation

- Stage One: Process Alerts
- Stage Two: Load TF Model for Inference

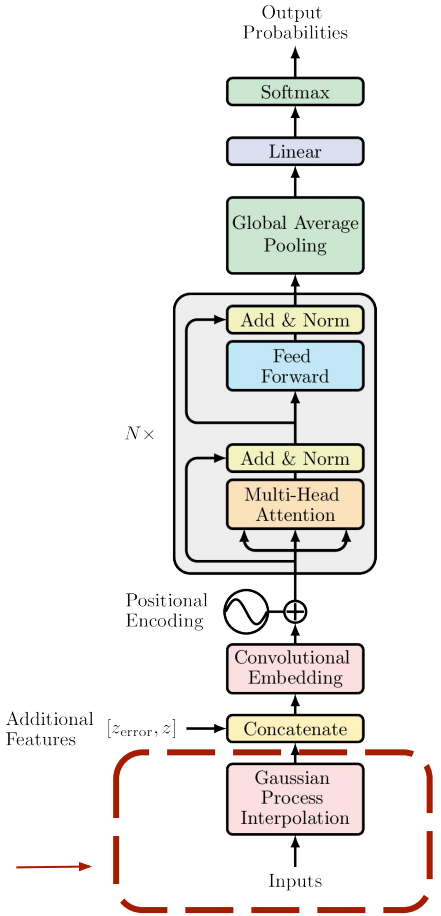
`fit_gps(raw_alert)` →



Stage One: Process Alert

Re-Format Raw Alert

```
>>> import pyspark.pandas as ps
>>> psdf = ps.read_parquet('sample.parquet')
>>> import random
>>> r = random.randint(0,len(psdf))
>>> alert = psdf.iloc[r]
>>> print(alert.head())
candid                                    1786552611115010001
schemavsn                                                 3.3
publisher                                                Fink
objectId                                            ZTF18aaqfhlj
candidate    (2459541.0526157, 2, 1786552611115, 19.1966800...
Name: 221, dtype: object
>>> alert = alert.to_dict()

>>> from fink_client.visualisation import extract_field
# Get flux and error
>>> magpsf = extract_field(alert, 'magpsf')
>>> sigmapsf = extract_field(alert, 'sigmapsf')

>>> jd = extract_field(alert, "jd")

# For rescaling dates to start at 0 --> 30
# dates = np.array([jd[0] - i for i in jd])

# FINK candidate ID (int64)
>>> candid = alert["candid"]

# filter bands
>>> fid = extract_field(alert, "fid")
```



`fit_gps(raw_alert)` →

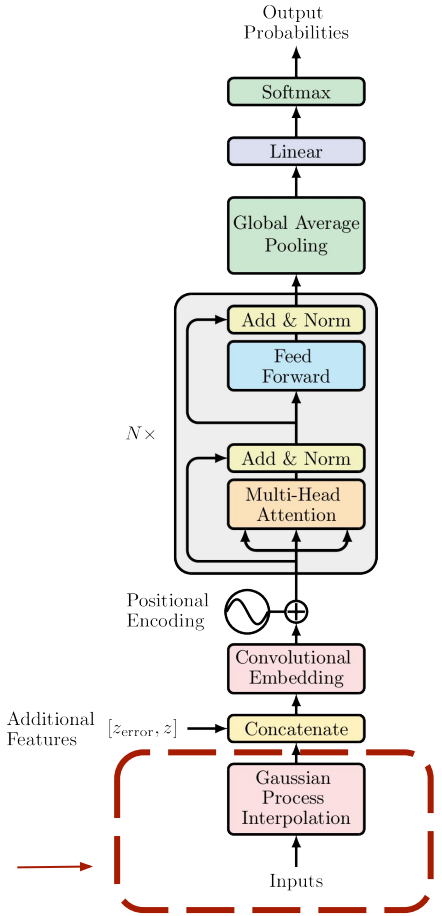
Stage One: Process Alert

Re-Format Raw Alert

```
>>> import pyspark.pandas as ps
>>> psdf = ps.read_parquet('sample.parquet')
>>> import random
>>> r = random.randint(0,len(psdf))
>>> alert = psdf.iloc[r]
>>> print(alert.head())
candid                                    1786552611115010001
schemavsn                                                 3.3
publisher                                                Fink
objectId                                           ZTF18aaqfhlj
candidate    (2459541.0526157, 2, 1786552611115, 19.1966800...
Name: 221, dtype: object
>>> alert = alert.to_dict()

>>> from fink_client.visualisation import extract_field
# Get flux and error
>>> magpsf = extract_field(alert, 'magpsf')
>>> sigmapsf = extract_field(alert, 'sigmapsf')

>>> jd = extract_field(alert, "jd")

# For rescaling dates to start at 0 --> 30
# dates = np.array([jd[0] - i for i in jd])

# FINK candidate ID (int64)
>>> candid = alert["candid"]

# filter bands
>>> fid = extract_field(alert, "fid")
```

object_id	mjd	flux	flux_error	filter
ZTF18abjrdau	27.9536	16.3509	0.101793	ztfg
ZTF18abjrdau	18.0079	16.7492	0.072451	ztfg
ZTF18abjrdau	16.0102	16.2887	0.09333	ztfg
ZTF18abjrdau	13.9637	16.3082	0.082149	ztfg
ZTF18abjrdau	8.95426	16.4864	0.067591	ztfg

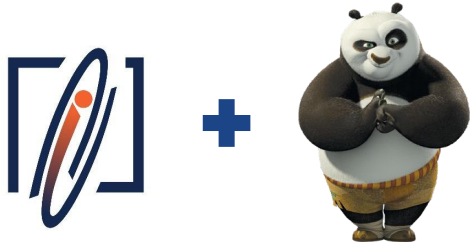
`fit_gps(raw_alert)` →



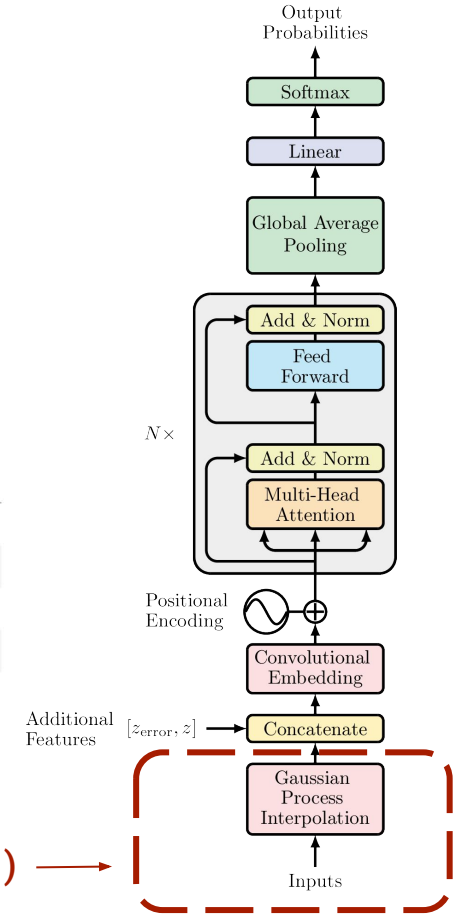
Stage One: Process Alert

Re-Format Raw Alert

```
>>> import pyspark.pandas as ps
>>> psdf = ps.read_parquet('sample.parquet')
>>> import random
>>> r = random.randint(0,len(psdf))
>>> alert = psdf.iloc[r]
>>> print(alert.head())
candid                                    1786552611115010001
schemavsn                                                 3.3
publisher                                                Fink
objectId                                           ZTF18aaqfhlj
candidate    (2459541.0526157, 2, 1786552611115, 19.1966800...
Name: 221, dtype: object
>>> alert = alert.to_dict()

>>> from fink_client.visualisation import extract_field
# Get flux and error
>>> magpsf = extract_field(alert, 'magpsf')
>>> sigmapsf = extract_field(alert, 'sigmapsf')

>>> jd = extract_field(alert, "jd")

# For rescaling dates to start at 0 --> 30
# dates = np.array([jd[0] - i for i in jd])

# FINK candidate ID (int64)
>>> candid = alert["candid"]

# filter bands
>>> fid = extract_field(alert, "fid")
```



object_id	mjd	flux	flux_error	filter
ZTF18abjrdau	27.9536	16.3509	0.101793	ztfg
ZTF18abjrdau	18.0079	16.7492	0.072451	ztfg
ZTF18abjrdau	16.0102	16.2887	0.09333	ztfg
ZTF18abjrdau	13.9637	16.3082	0.082149	ztfg
ZTF18abjrdau	8.95426	16.4864	0.067591	ztfg

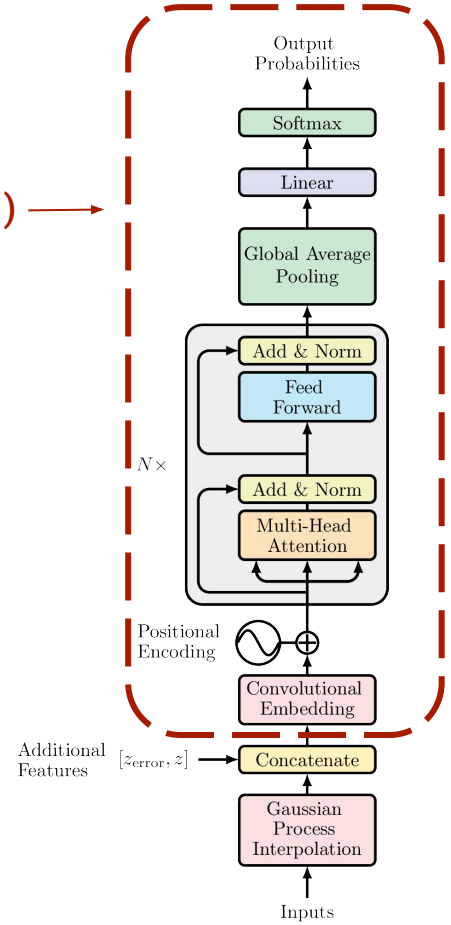


`fit_gps(raw_alert)` →

Stage Two: Black Box Inference

Load Pre-trained Model

```
model.predict(processed_alert)
```

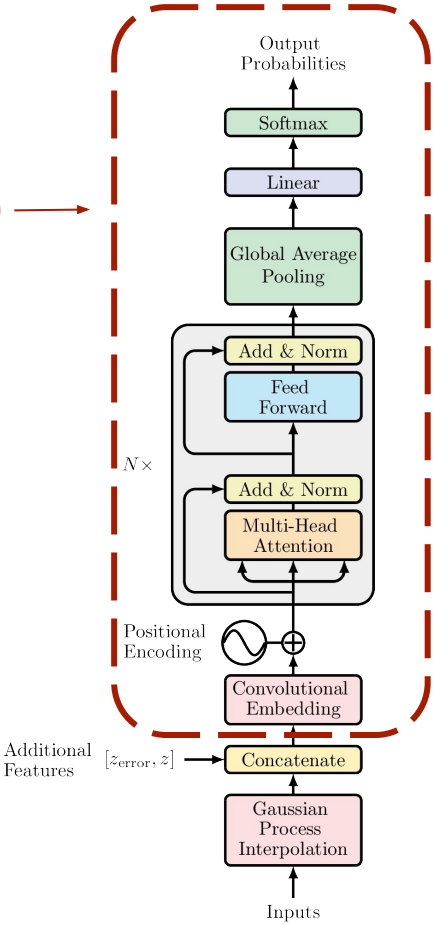


Stage Two: Black Box Inference

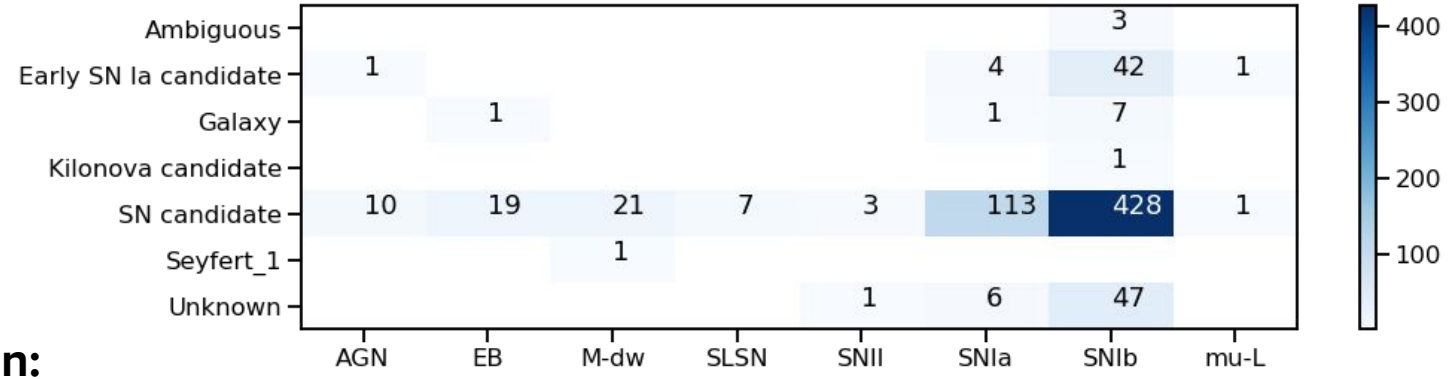
Load Pre-trained Model

```
model.predict(processed_alert)
```

```python
def get_model(model_name: str = 't2', model_id: str = "23057-1642540624-0.1.dev963+g309c9d8"):
    """ Load pre-trained model for T2

    Parameters
    ----------
    model_name: str
        Folder name containing pre-trained models. Available: t2, atx
    model_id: str
        Corresponding ID inside the foler (related to the version used to train)

    Returns
    ----------
    out: keras model
    """
    model_path = (
        f"{Path(__name__).absolute().parent.parent}/data/models/{model_name}/model-{model_id}"
    )

    model = keras.models.load_model(
        model_path,
        custom_objects={"WeightedLogLoss": WeightedLogLoss()},
        compile=False,
    )

    return model
```



Performance, *thus far …*

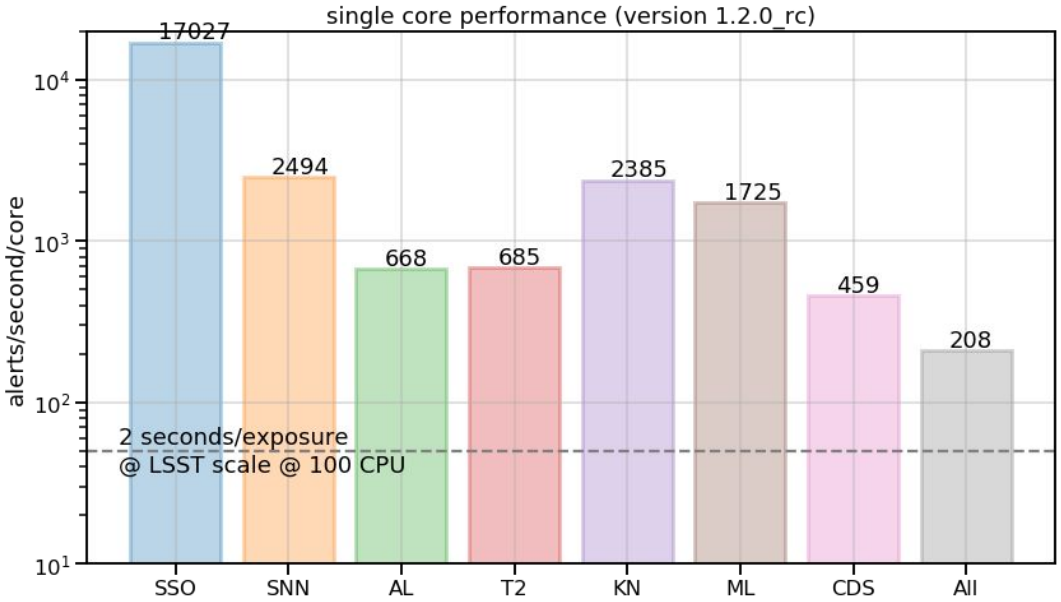


Trial Run:

- 161,462 –> 715 alerts (one full night) after selection cuts, compare “max prob” class from t2 with FINK scoring.
- New predictions for “Unknown”, but seems to be a bias for SNIb.
- TNS validation does well for SN in general, but confirms a slight bias to SNIb instead of SNIa.

Performance, *thus far …*

- Still a few tweaks that could improve throughput
- Need to better understand t2's *value add,* max_prob vs all probs?
- Where should t2 sit along the pipeline?

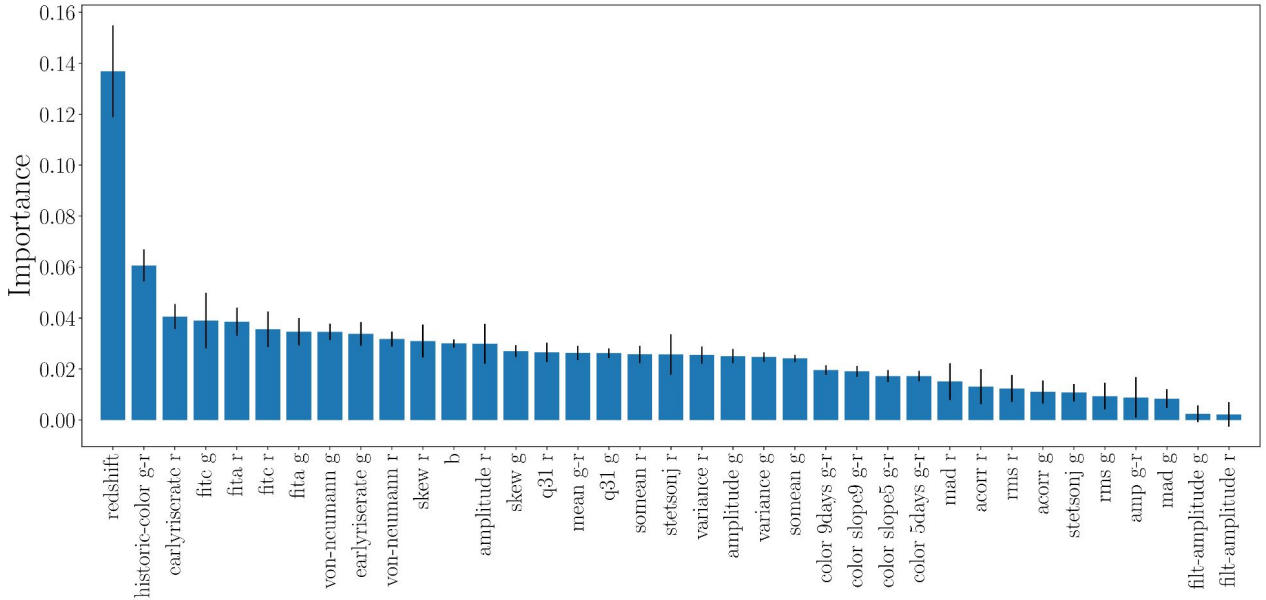


Discussion...

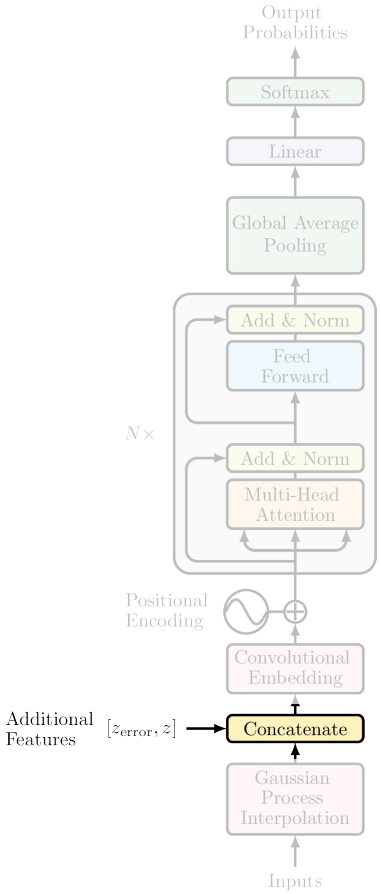
APPENDIX

Inputting Addition Information

- Domain knowledge still important!



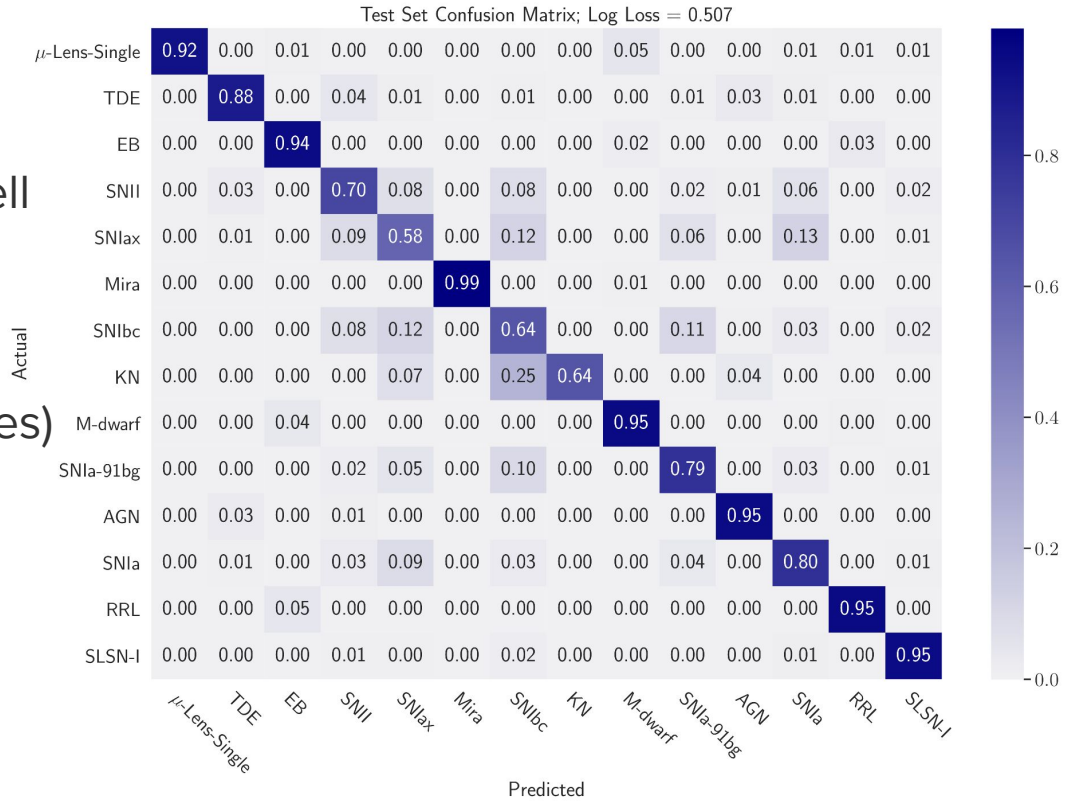
*Boone et al. 2019



Performance and Results (PLAsTiCC: *ugrizy+z*)

Confusion Matrix

- Handles class imbalance well
- Log-Loss **0.507** (with *z*)
- Log-Loss **0.8** (only time-series)
- Should be noted this is in a *representative* setting



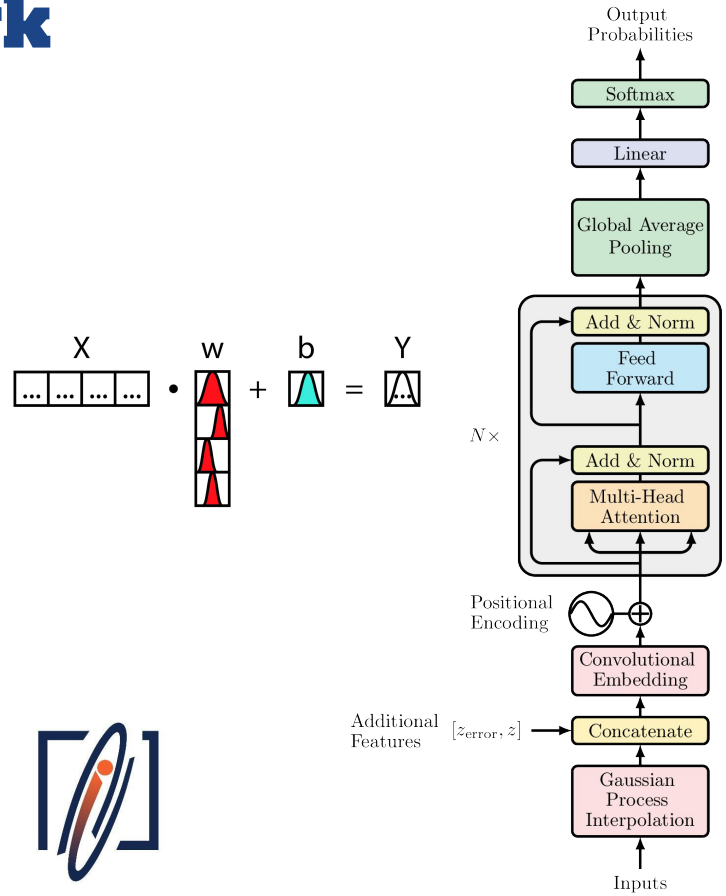
Extensions and Future Work

Extending Time-Series Transformer [t2]

- Inclusion of Decoder-block for early light curve classification
- Probabilistic extensions for better UQ

Future Work

- Test on real data
- Put into production in real-time setting



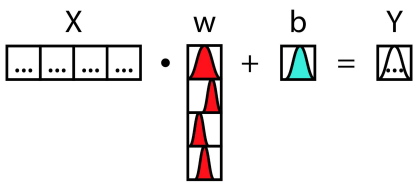
Extensions and Future Work

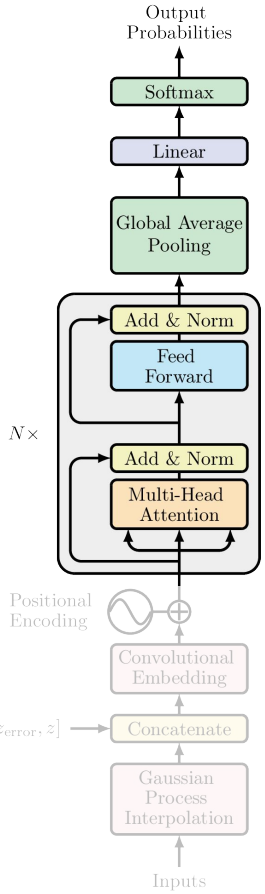
Extending Time-Series Transformer [t2]

- Inclusion of Decoder-block for early light curve classification
- Probabilistic extensions for better UQ

Future Work

- Test on real data
- Put into production in real-time setting

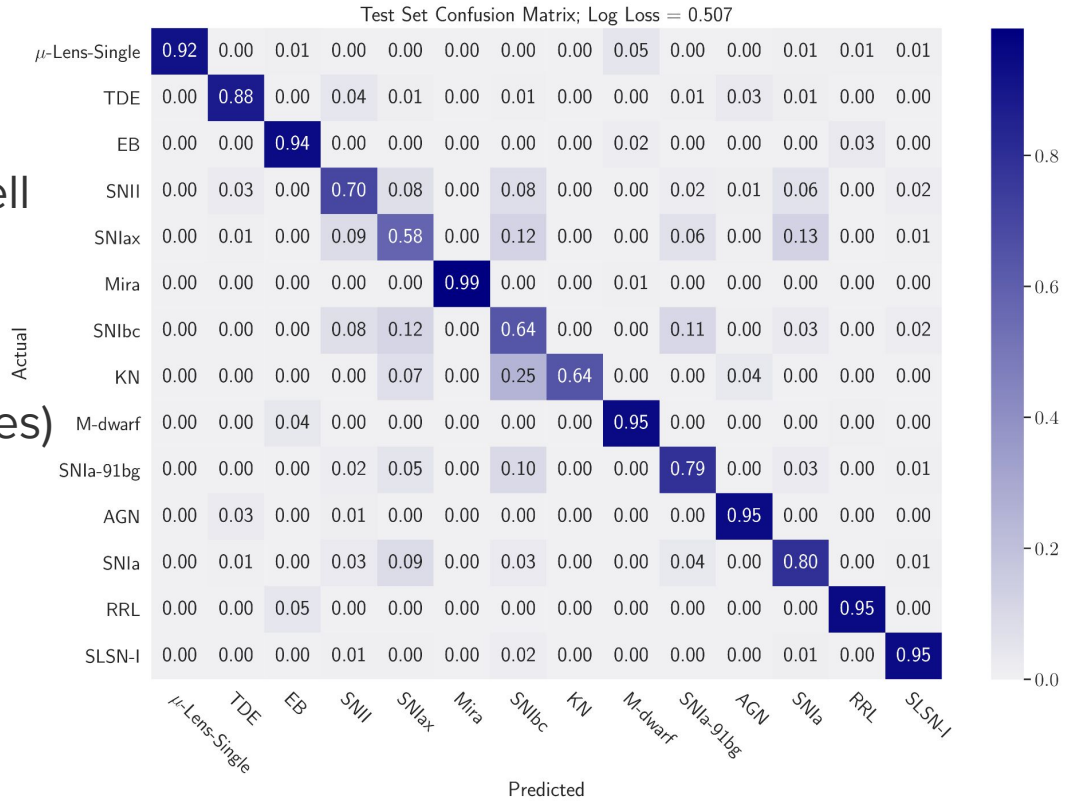




Performance and Results

Confusion Matrix

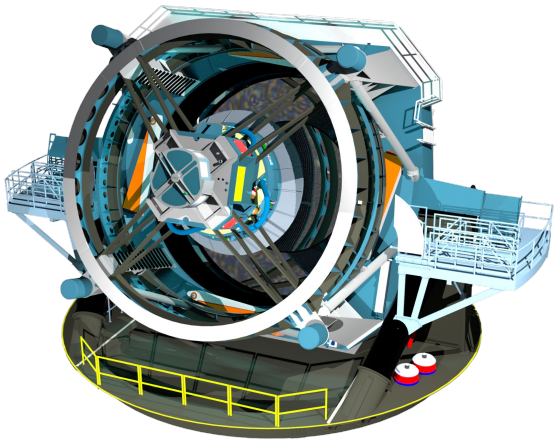
- Handles class imbalance well
- Log-Loss **0.507** (with *z*)
- Log-Loss **0.8** (only time-series)
- Should be noted this is in a *representative* setting



Motivation: A Deluge of Data

Overview

- 10 million alerts, per night!
- Machine Learning methods are now critical
- Accurate and fast classification required for follow up
- Desire for UQ and interpretability when allocating follow up resources.



Deep Learning to the Rescue!?

The death of feature engineering?

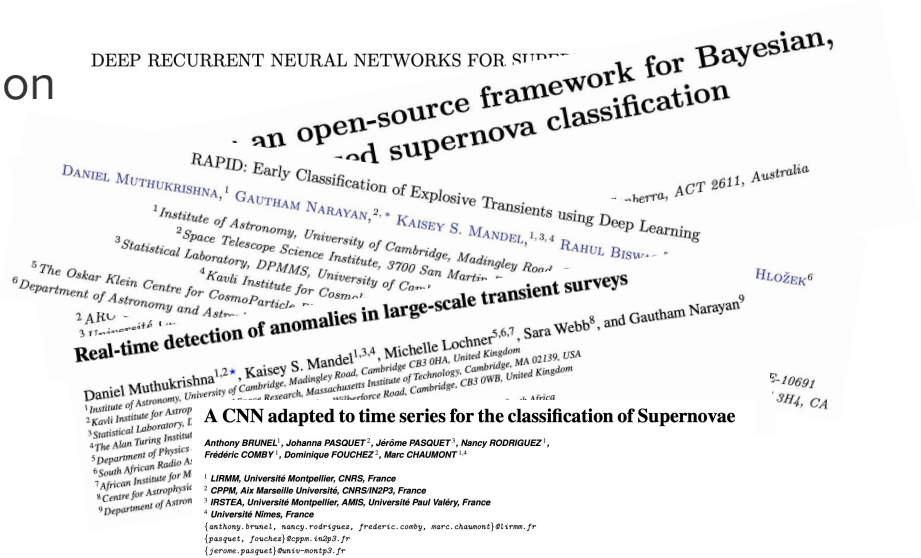
- Exploiting inherent time-series information

RNNs (inc. LSTMs, GRUs)

- Hard to train, not easily parallelizable
- Large memory footprint

CNNs (inc. TCN)

- Computationally expensive


DEEP RECURRENT NEURAL NETWORKS FOR SUP

an open-source framework for Bayesian,
od supernova classification

RAPID: Early Classification of Explosive Transients using Deep Learning

DANIEL MUTHUKRISHNA,[1] GAUTHAM NARAYAN,[2,*] KAISEY S. MANDEL,[1,3,4] RAHUL BISW

[1] Institute of Astronomy, University of Cambridge, Madingley Road, Cambridge CB3 0HA, United Kingdom
[2] Space Telescope Science Institute, 3700 San Martin
[3] Statistical Laboratory, DPMMS, University of Cam
[4] Kavli Institute for Cosm
[5] The Oskar Klein Centre for CosmoParticle
[6] Department of Astronomy and Astr
[7] ARC
[8] Université

herra, ACT 2611, Australia

HLOŽEK[6]

Real-time detection of anomalies in large-scale transient surveys

Daniel Muthukrishna[1,2,*], Kaisey S. Mandel[1,3,4], Michelle Lochner[5,6,7], Sara Webb[8], and Gautham Narayan[9]

[1] Institute of Astronomy, University of Cambridge, Madingley Road, Cambridge CB3 0HA, United Kingdom
[2] Kavli Institute for Astrop
[3] Statistical Laboratory, I
[4] The Alan Turing Institut
[5] Department of Physics
[6] South African Radio A
[7] African Institute for M
[8] Centre for Astrophysic
[9] Department of Astron
Research, Massachusetts Institute of Technology, Cambridge, MA 02139, USA
Cambridge, CB3 0WB, United Kingdom
Africa

5-10691
3H4, CA

A CNN adapted to time series for the classification of Supernovae

Anthony BRUNEL[1], Johanna PASQUET[2], Jérôme PASQUET[3], Nancy RODRIGUEZ[1], Frédéric COMBY[1], Dominique FOUCHEZ[2], Marc CHAUMONT[1,4]

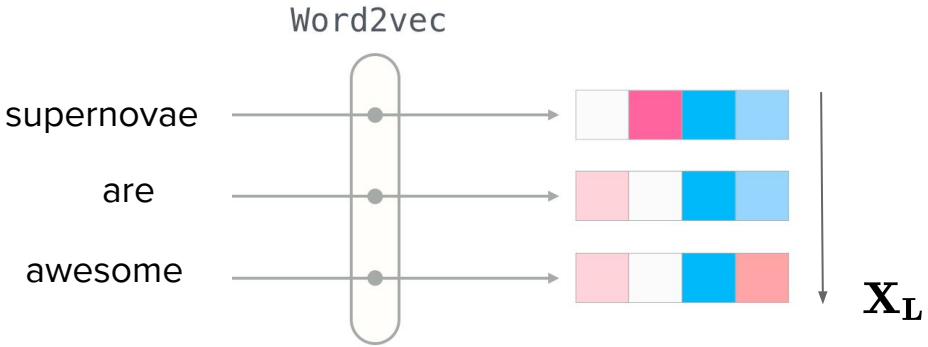
[1] LIRMM, Université Montpellier, CNRS, France
[2] CPPM, Aix Marseille Université, CNRS/IN2P3, France
[3] IRSTEA, Université Montpellier, AMIS, Université Paul Valéry, France
[4] Université Nîmes, France

{anthony.brunel, nancy.rodrigues, frederic.comby, marc.chaumont}@lirmm.fr
{pasquet, fouchez}@cppm.in2p3.fr
{jerome.pasquet}@univ-montp3.fr

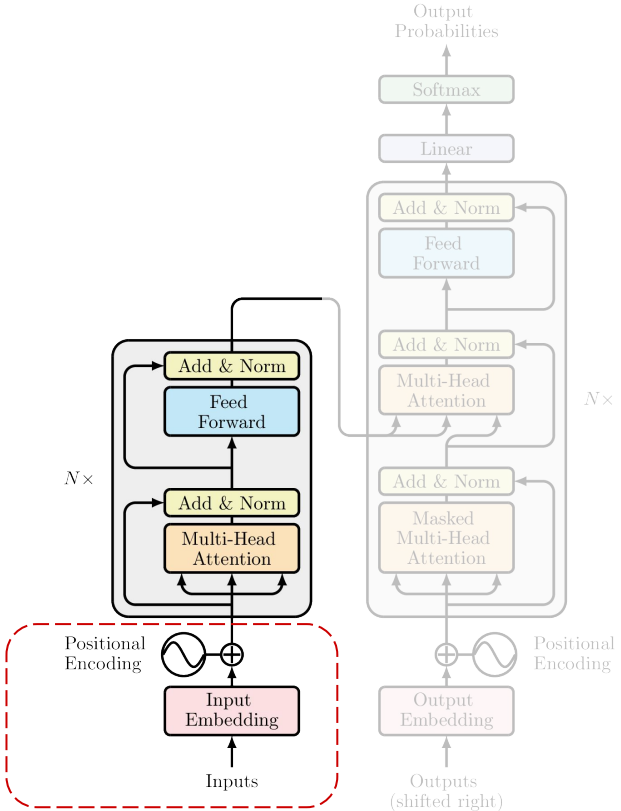
Input Embedding and Positional Encoding

Vector space representations

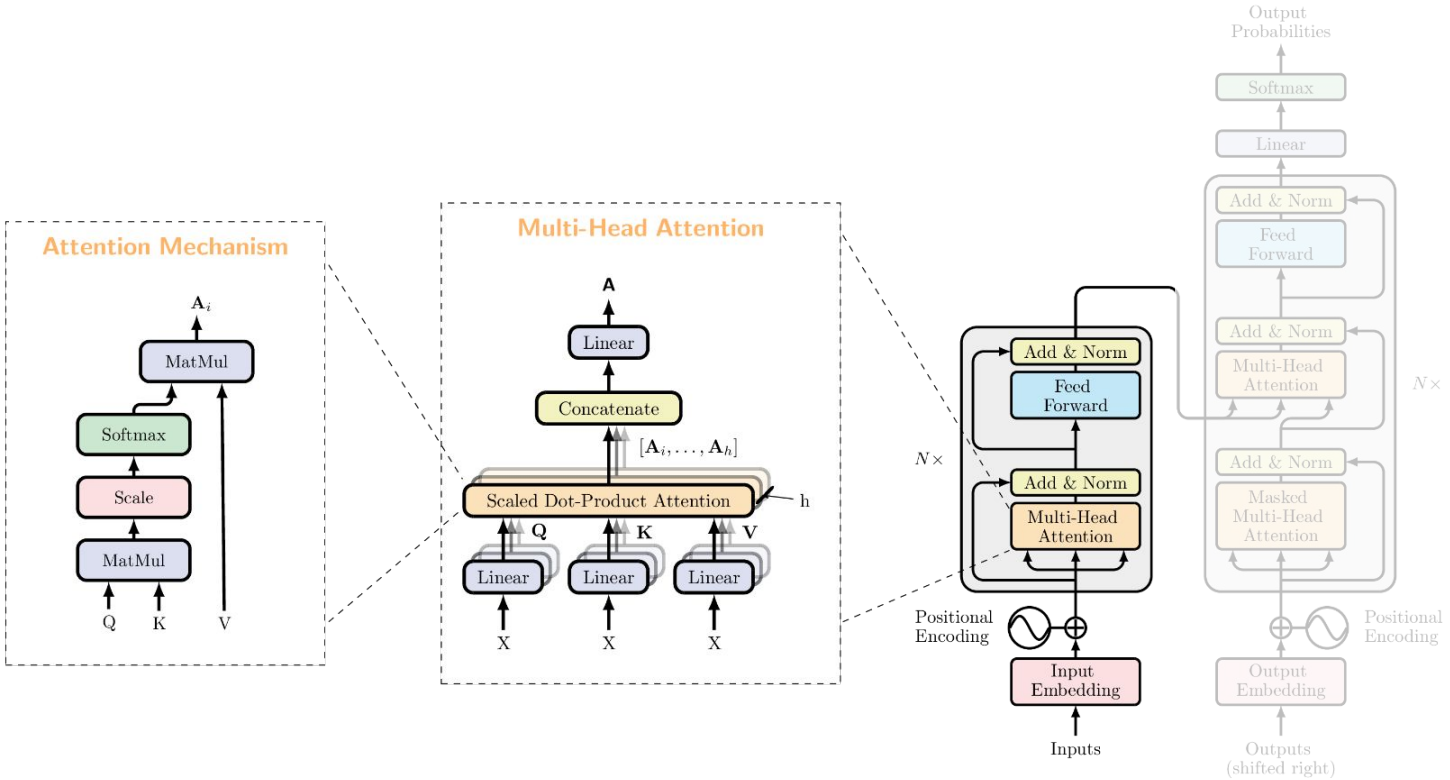
"supernovae are awesome"



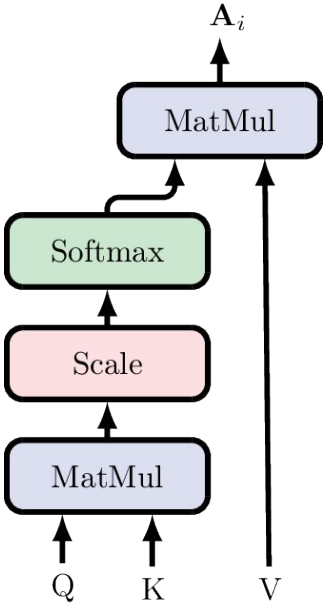
$$\boldsymbol{x}_l \leftarrow \boldsymbol{x}_l \oplus \boldsymbol{p}_l .$$



Multi-headed Self Attention



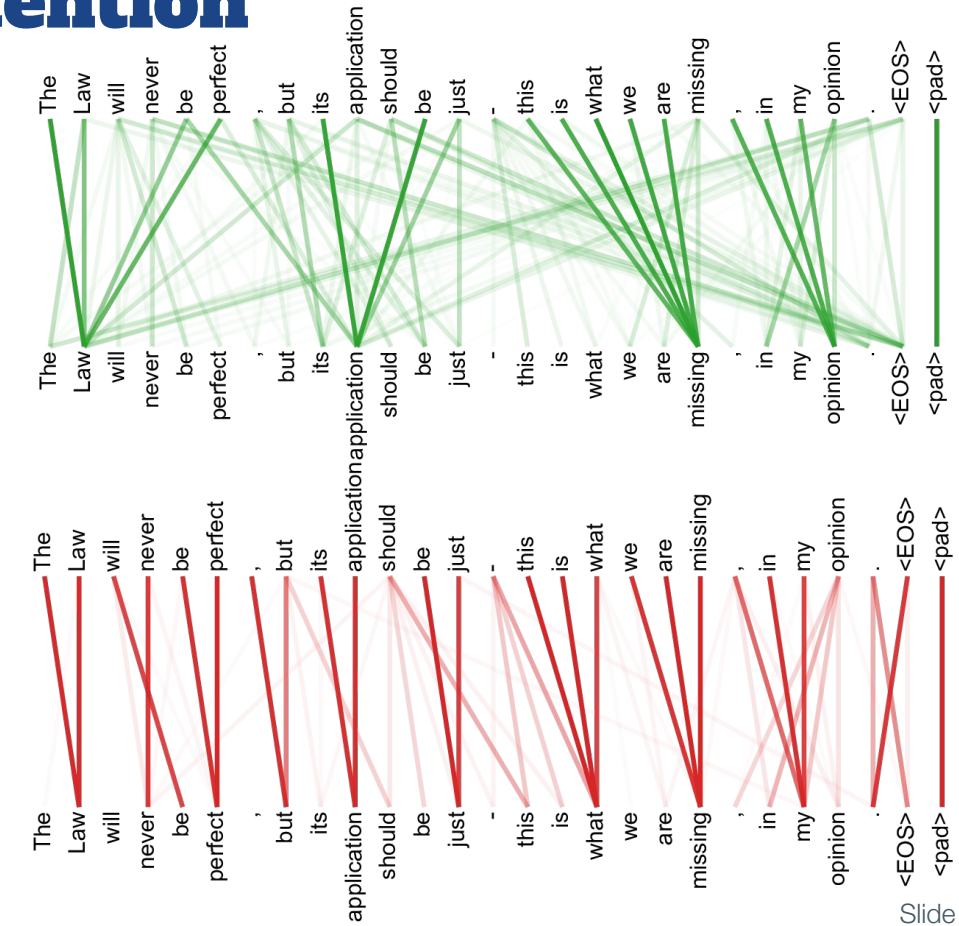
Multi-headed Self Attention



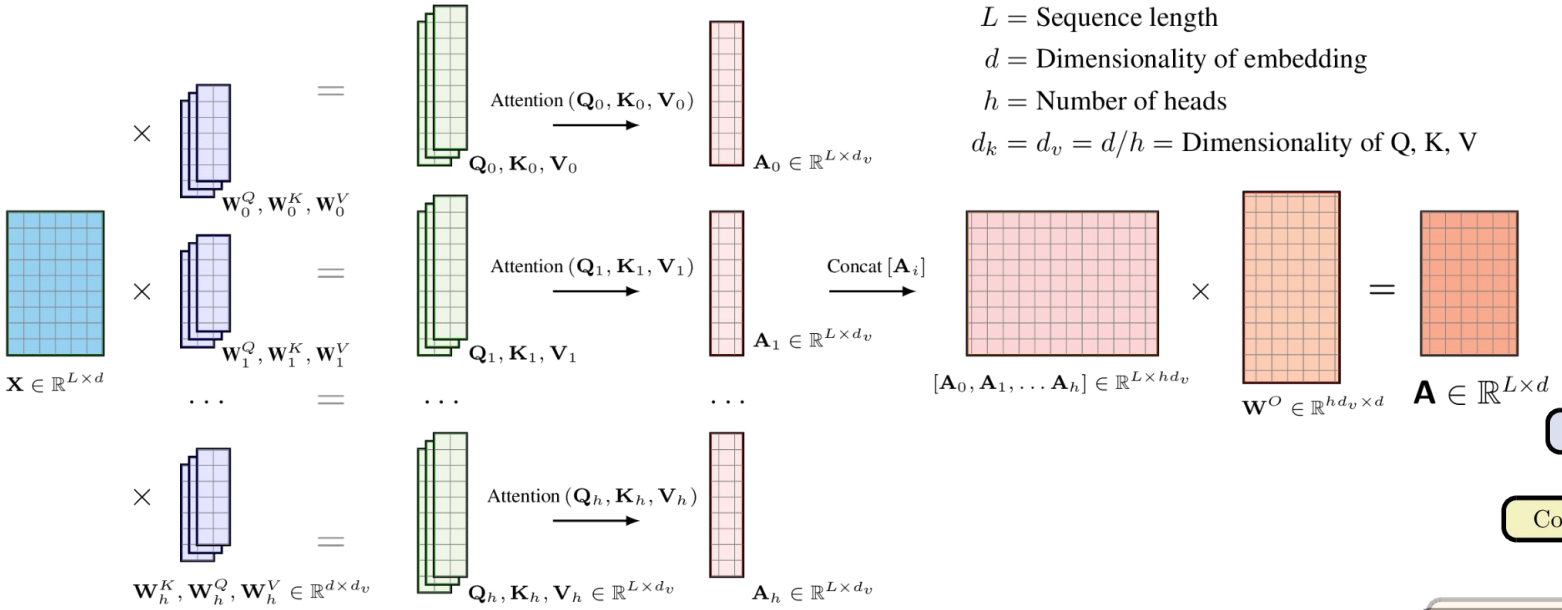
$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathbf{A} = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V}.$$

$$\mathbf{Q} = \mathbf{X}\mathbf{W}^Q \in \mathbb{R}^{L \times d_q},$$

$$\mathbf{K} = \mathbf{X}\mathbf{W}^K \in \mathbb{R}^{L \times d_k}, \mathbf{V} = \mathbf{X}\mathbf{W}^V \in \mathbb{R}^{L \times d_v}$$



Multi-headed Self Attention

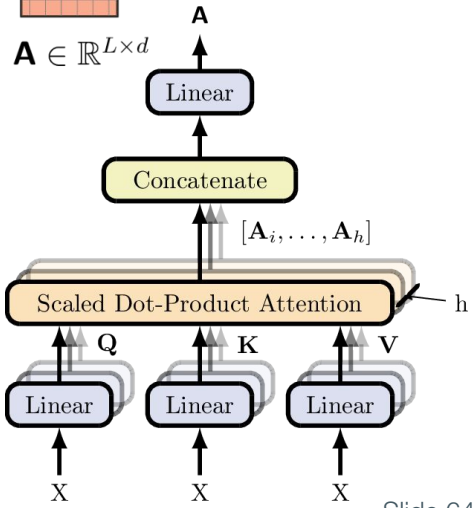


L = Sequence length

d = Dimensionality of embedding

h = Number of heads

$d_k = d_v = d/h$ = Dimensionality of Q, K, V

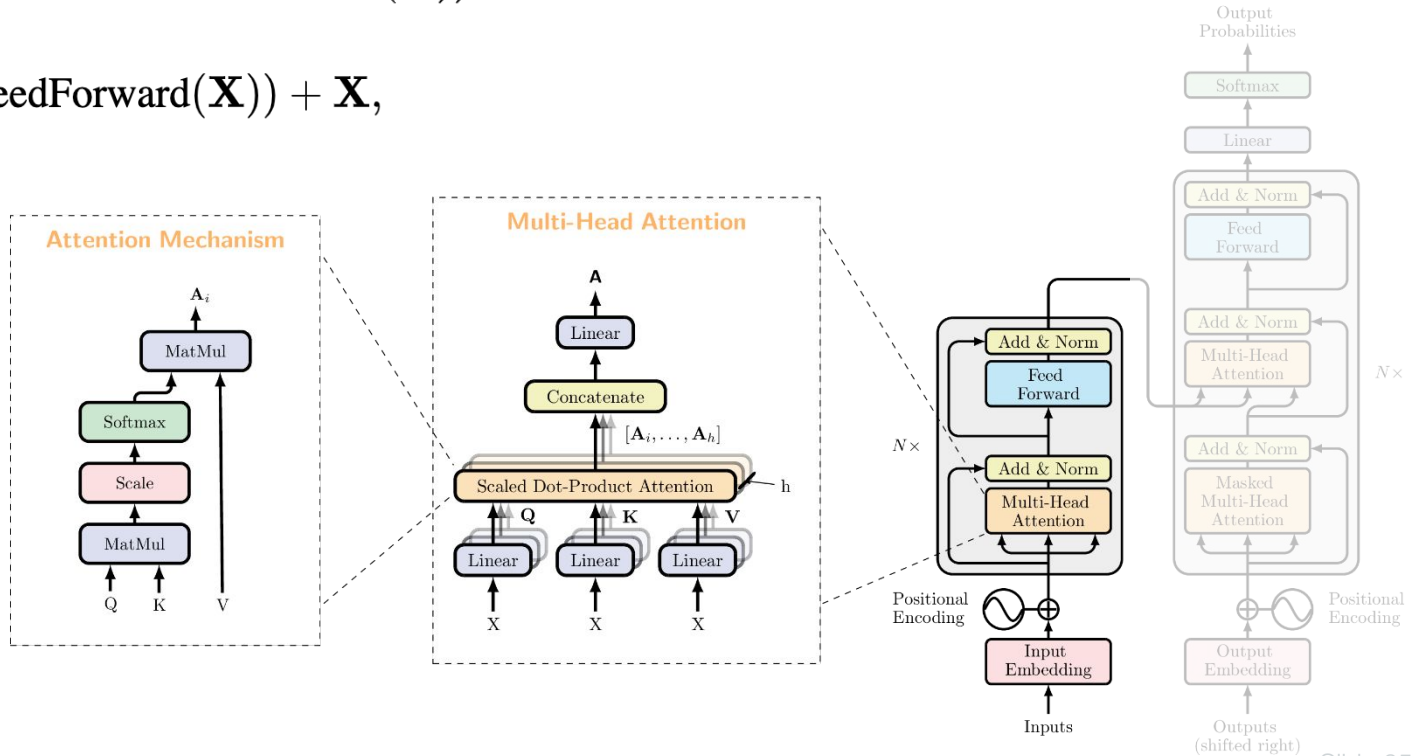


$$\text{MultiHead}\left(\mathbf{Q}, \mathbf{K}, \mathbf{V}\right) = \mathbf{A} = \text{Concat}[\mathbf{A}_1, \ldots, \mathbf{A}_h]\mathbf{W}^O,$$

A Transformer Block [Encoder]

1. $\mathbf{X} \leftarrow \text{LayerNorm}(\text{MultiHeadSelfAttention}(\mathbf{X})) + \mathbf{X}.$

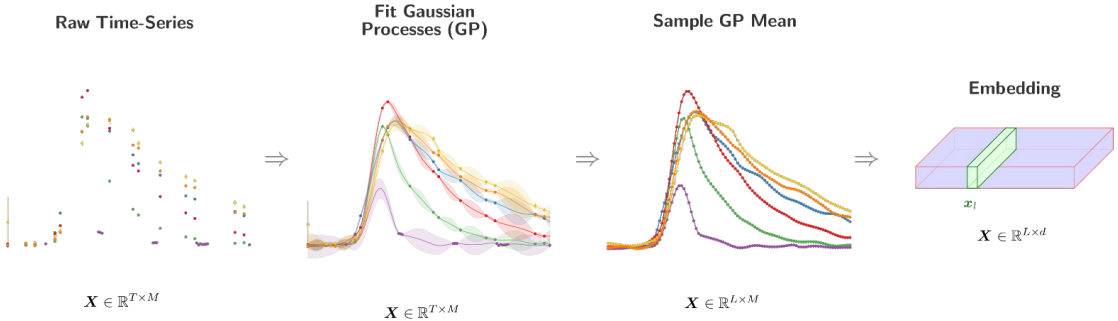
2. $\mathbf{X} \leftarrow \text{LayerNorm}(\text{FeedForward}(\mathbf{X})) + \mathbf{X},$

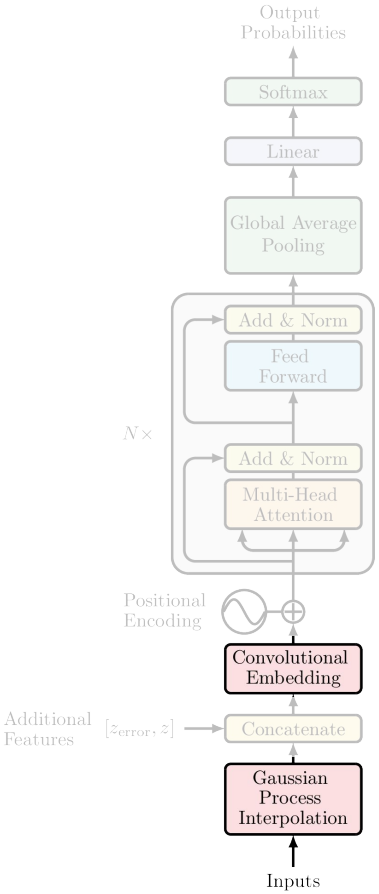


Convolutional Embedding

Projecting photometric data into vector spaces

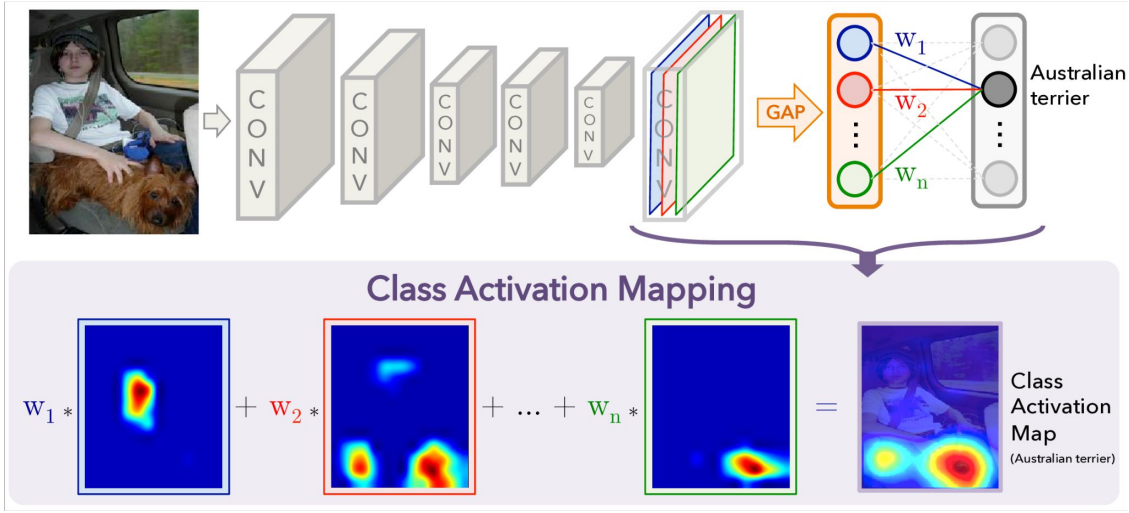
- Think of `timestep2vec` akin to `word2vec`
- 1-dimensional convolution allows for scaling of M → d
- Positional encoding follows as before

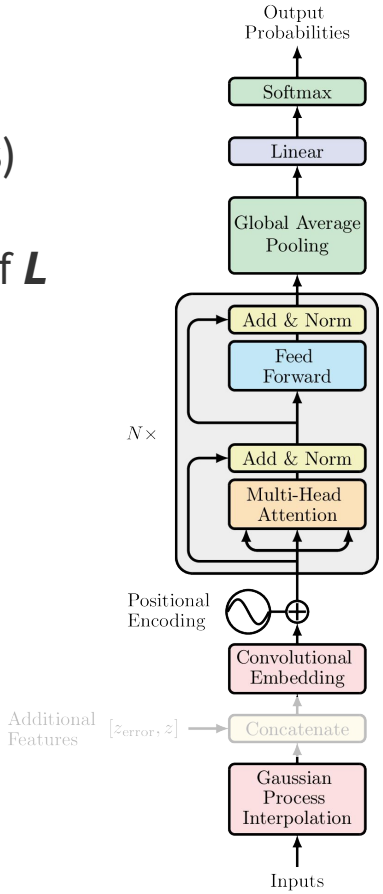




Global Average Pooling (GAP)

- Inclusion of GAP allows for Class Activation Maps (CAMs)
- CAMs adapted for use with time-series, i.e. as function of ***L***

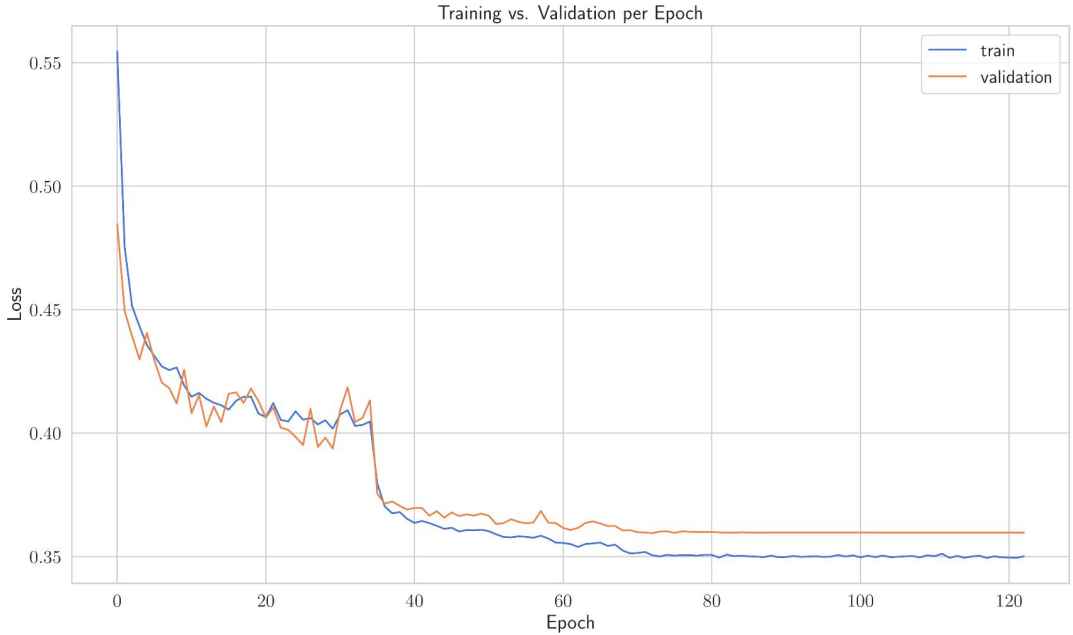




Performance and Results

Training and Inference

- PLAsTiCC dataset ~ 3M LCs
- Imbalanced, representative
- Extensibility for multi-GPU or TPU
- Low #params for fast inference*

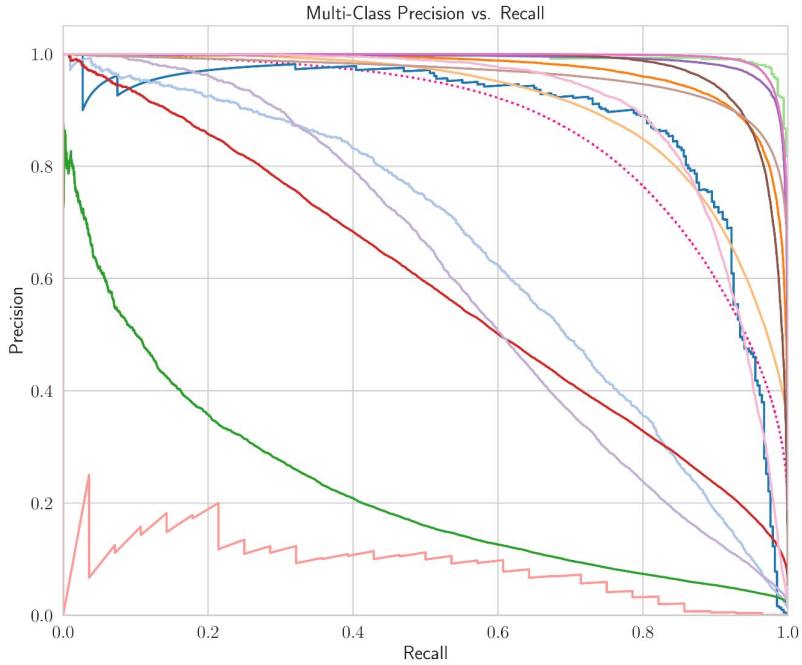


*does not include GP interpolation step

Performance and Results

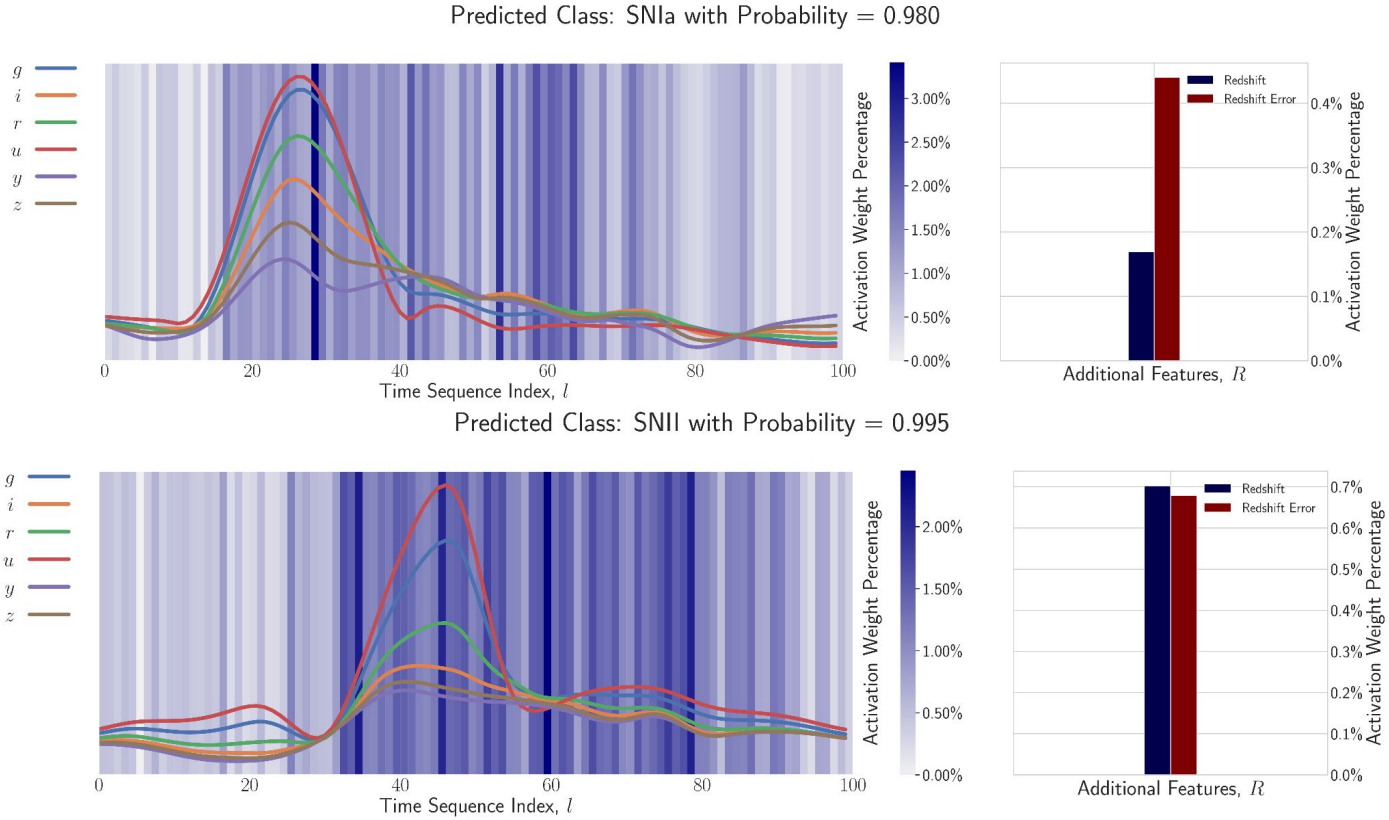
Precision-Recall

- AUC ~ **0.87**
- KNe struggling (0.004%)
- SNIax mistaken for Type Ia?



Performance and Results

CAMs



Questions?