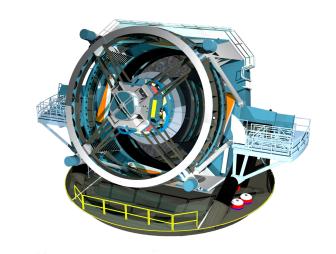
Low-latency High-throughput Classification using Deep Model Compression

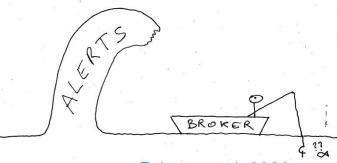


Motivation: A Deluge of Data

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)

RAPID: Early Classification of Explosive Transients using Deep Learning Daniel Muthukrishna, ¹ Gautham Narayan, ². * Kaisey S. Mandel, ^{1,3,4} Rahul Biswee ¹Institute of Astronomy, University of Cambridge, Madingley Road Space Telescope Science Institute, 3700 San Martin ³ Statistical Laboratory, DPMMS, University of Cam. ⁵The Oskar Klein Centre for CosmoParticle Real-time detection of anomalies in large-scale transient surveys HLOŽEK6 ⁶Department of Astronomy and Astronomy Daniel Muthukrishna 1.2*, Kaisey S. Mandel 1.34, Michelle Lochner 5.67, Sara Webb⁸, and Gautham Narayan of the state of Institute of Astronomy, University of Cambridge, Madingley Road, Cambridge CBS 0HA, United Kingdom

Paralle of Astronomy, University of Cambridge, Madingley Road, Cambridge CBS 0HA, United Kingdom

Paralle Institute of Astronomy, University of Cambridge, Madingley Road, Cambridge CBS 0HA, United Kingdom

Paralle of CBS 0HA, United Kingdom

Pa og camaringe, menungey som, camaringe cas orac, camaringe, MA 02130, USA
Research Massachuserts Institute of Technolosy, Cambridge, MA 02130, USA
NN adoors.

NN adoors. E-10691 3H4, CA ²Kavli Institute for Astrop A CNN adapted to time series for the classification of Supernovae 3 Statistical Laboratory, L ⁴The Alan Turing Institut Anthony BRUNEL¹, Johanna PASQUET², Jérôme PASQUET³, Nancy RODRIGUEZ¹ Frédéric COMBY 1, Dominique FOUCHEZ 2, Marc CHAUMONT 1,4 LIRMM, Université Montpellier, CNRS, France ² CPPM, Aix Marseille Université, CNRS/IN2P3, France 9 Department of Astron 3 IRSTEA, Université Montpellier, AMIS, Université Paul Valéry, France ⁴ Université Nimes France {anthony.brunel, mancy.rodrigues, frederic.comby, marc.chaumont}@lirmm.fr {pasquet, fouches}@cppm.in2p3.fr

{ jerome.pasquet}@univ-montp3.fr

DEEP RECURRENT NEURAL NETWORKS FOR SUPER-

an open-source framework for Bayesian,

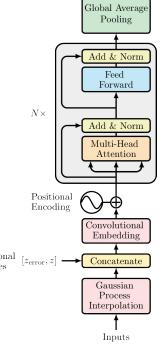
and supernova classification

nherra, ACT 2611, Australia

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

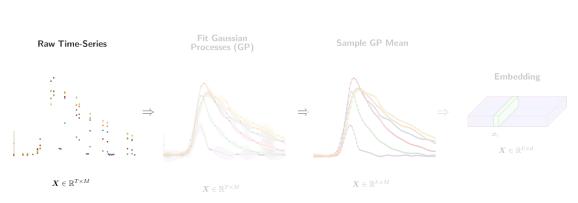


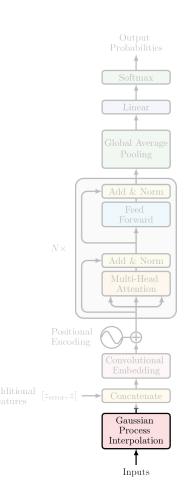
Output Probabilities

Softmax

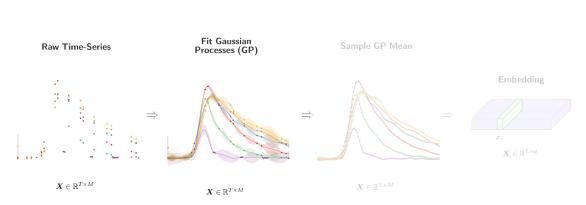
Linear

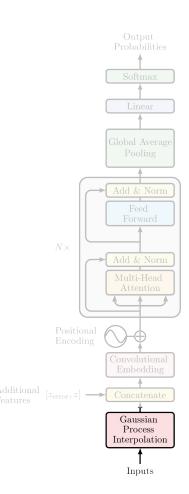
- 2D-Matern kernel for Gaussian Process interpolation
- Evaluate at regular period, 100 points in our case



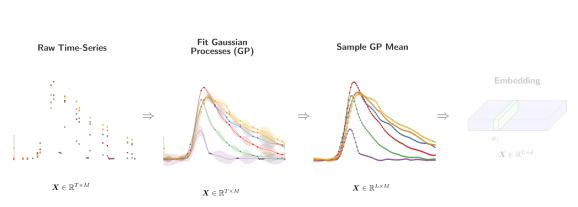


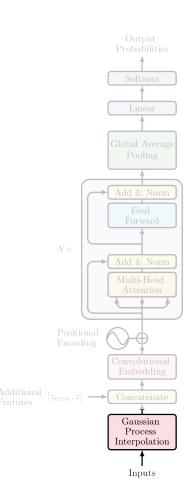
- 2D-Matern kernel for Gaussian Process interpolation
- Evaluate at regular period, 100 points in our case



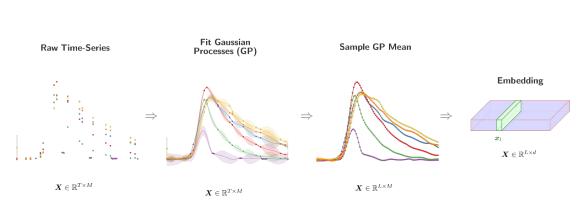


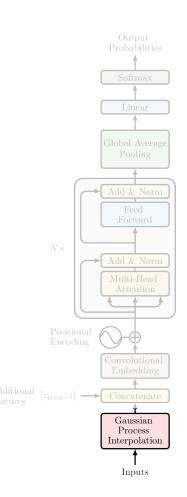
- 2D-Matern kernel for Gaussian Process interpolation
- Evaluate at regular period, 100 points in our case





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

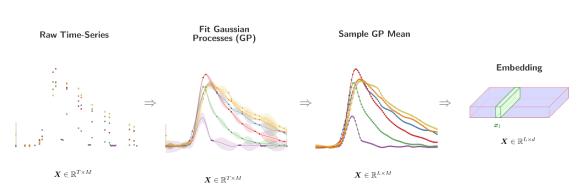


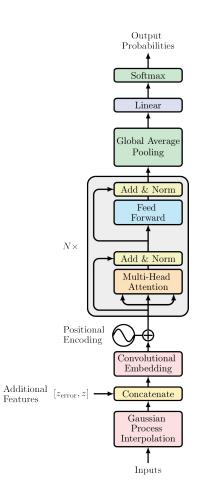


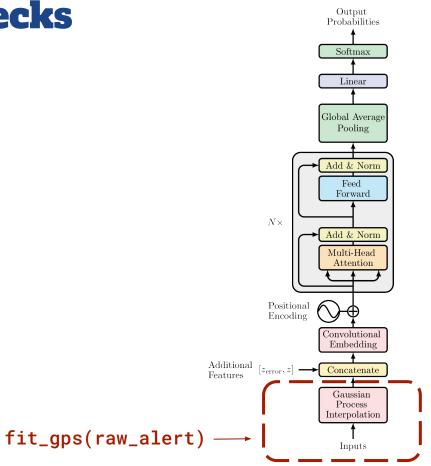
Gaussian Process Interpolation

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

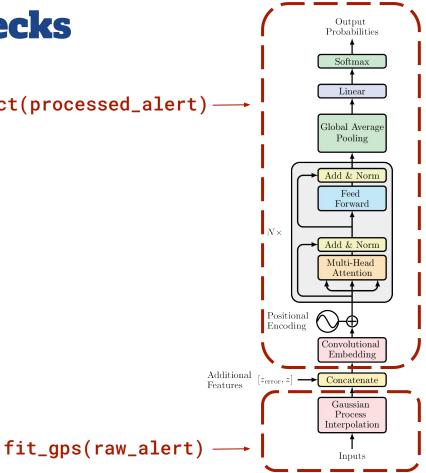
Deep Learning Magic ...







model.predict(processed_alert) -



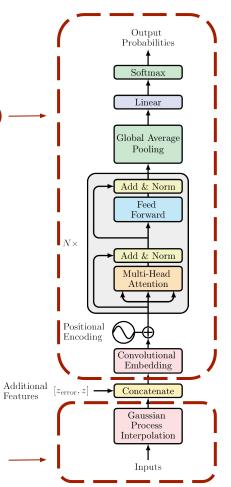
Total time: 5.85664 s File: get_models.py

model.predict(processed_alert) -

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	Tim	ne Per H	$it \ \ \ Ti$	me Line Contents	
206	16	139.6	8.7	9.5	df_gp_mean = generate_gp_all_objects()	
• • •						
• • •						
212 213	8 8	180.8 12.3	22.6 1.5		<pre>X = df_gp_mean[cols] X = rs(X)</pre>	
		12.0	1.0			Addi Featı
217	8	1101.7	137.7	74.9	y_preds = model.predict(X)	. \
					fit_gps(raw_alert	:) —





Total time: 5.85664 s
File: get_models.py

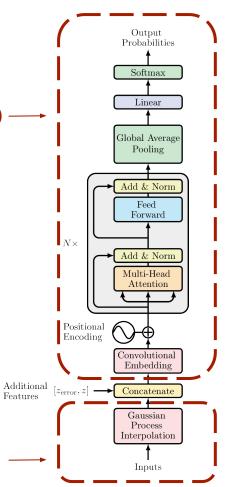
model.predict(processed_alert) -

fit_gps(raw_alert) -

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	Tin	ne Per H	it \% Ti	me Line Contents
======					=======================================
206	16	139.6	8.7	9.5	<pre>df_gp_mean = generate_gp_all_objects()</pre>
* * *					
• • •					
212	8	180.8	22.6	12.3	<pre>X = df_gp_mean[cols]</pre>
213	8	12.3	1.5	0.8	X = rs(X)
217	8	1101.7	137.7	74.9	<pre>y_preds = model.predict(X)</pre>





Total time: 5.85664 s File: get_models.py

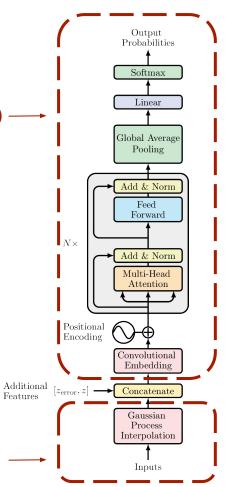
model.predict(processed_alert) -

fit_gps(raw_alert) -

Function: get_model at line 29

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

Line #	${\it Hits}$	Tim	ne Per H	it \% Tin	ne Line Contents
======			======		
• • •					
206	16	139.6	8.7	9.5	<pre>df_gp_mean = generate_gp_all_objects()</pre>
• • •					
• • •					
212	8	180.8	22.6	12.3	<pre>X = df_gp_mean[cols]</pre>
213	8	12.3	1.5	0.8	X = rs(X)
				()	
• • •					
217	8	1101.7	137.7	74.9	<pre>y_preds = model.predict(X)</pre>



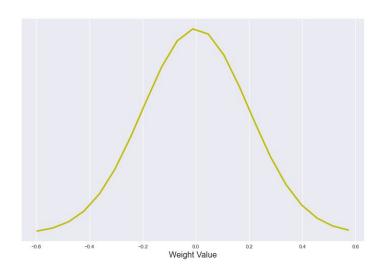
Deep Model Compression: An Almost Free Lunch

- 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

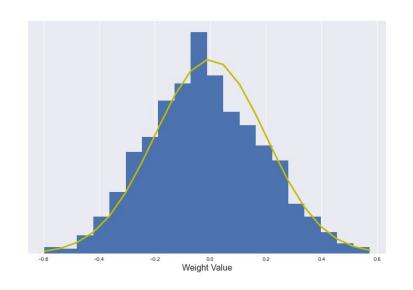
- Reduce number of unique weights with shared weight values
- Can then leverage Huffman encoding
- Note: best to avoid clustering early layers!



Deep Model Compression: Clustering

Weight Clustering

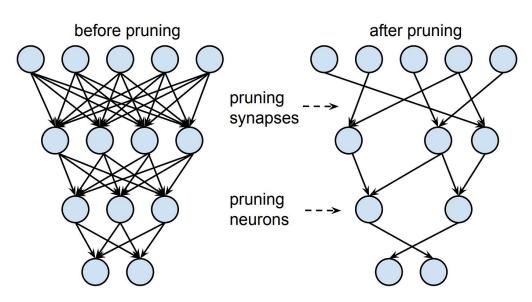
- Reduce number of unique weights with shared weight values
- Can then leverage Huffman encoding
- Note: best to avoid clustering early layers!



Deep Model Compression: Pruning

Weight Pruning

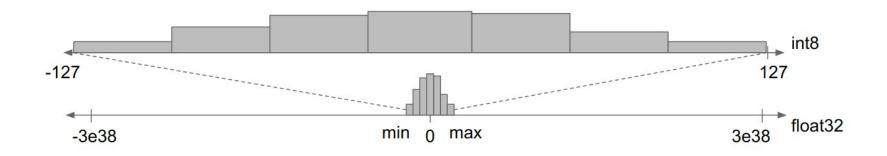
- Set low magnitude weights to zero
- Can then leverage Huffman encoding (again)
- Done during training



Deep Model Compression: Quantization

Weight Quantization

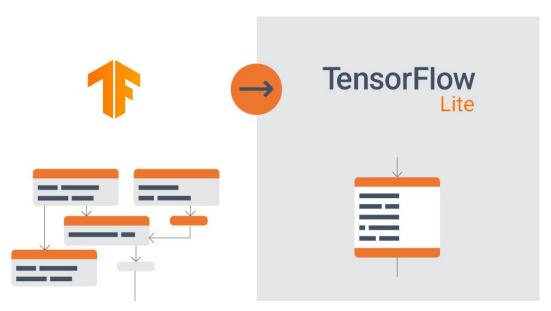
- Reduce precision of stored weight values
- Computations are still done at float32

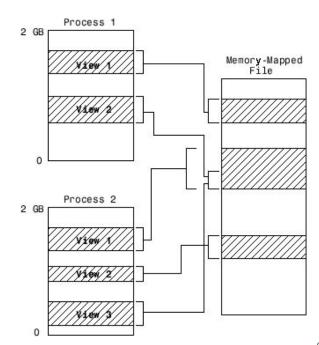


Efficient Frameworks & File Formats

From TensorFlow to TensorFlow-Lite

- Operator Fusion
- ProtocolBuffers → FlatBuffers





Compression Method	Model Size (kb)	Load Latency (s ⁻³)	Inference Latency (s)	Loss
BASELINE	1100	6324.145	0.333	0.968

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

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

				3
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

				- 5
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

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

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

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

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

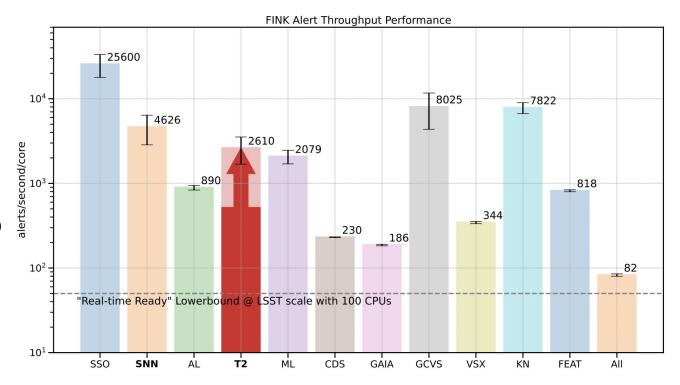
- **18** x reduction in model size
- 24,000 x load time improvement
- **8 x** inference latency improvement

Production Results

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

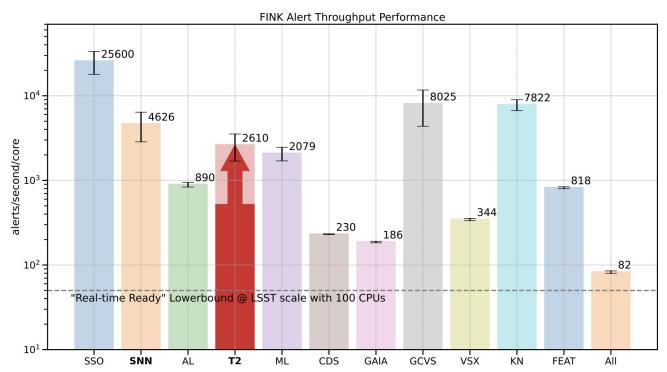
Production Results

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



Takeaways...

- Use of deep compression can be your friend, but be careful
- Use FlatBuffers where possible



Questions?