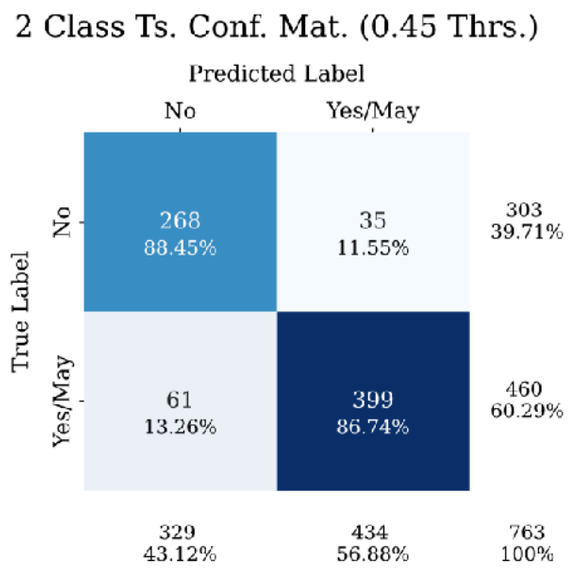
The Astronomical Journal, 169:159 (17pp), 2025 March
https://doi.org/10.3847/1538-3881/ada4b5

LensNet: Enhancing Real-time Microlensing Event Discovery with Recurrent Neural Networks in the Korea Microlensing Telescope Network

Javier Viaña[1,2,16], Kyu-Ha Hwang[3], Zoë de Beurs[1,2,4], Jennifer C. Yee[5], Andrew Vanderburg[1,2], Michael D. Albrow[6], Sun-Ju Chung[3], Andrew Gould[7,8], Cheongho Han[9], Youn Kil Jung[3,10], Yoon-Hyun Ryu[3], In-Gu Shin[5], Yossi Shvartzvald[11], Hongjing Yang[12], Weicheng Zang[5], Sang-Mok Cha[3,13], Dong-Jin Kim[3], Seung-Lee Kim[3], Chung-Uk Lee[3], Dong-Joo Lee[3], Yongseok Lee[3,13], Byeong-Gon Park[3], and Richard W. Pogge[14,15]

Recurrent Neural Networks (RNNs): application to KMTNet



Raphael Oliveira, 14.apr.2025, Statistics Journal Club, OAUW

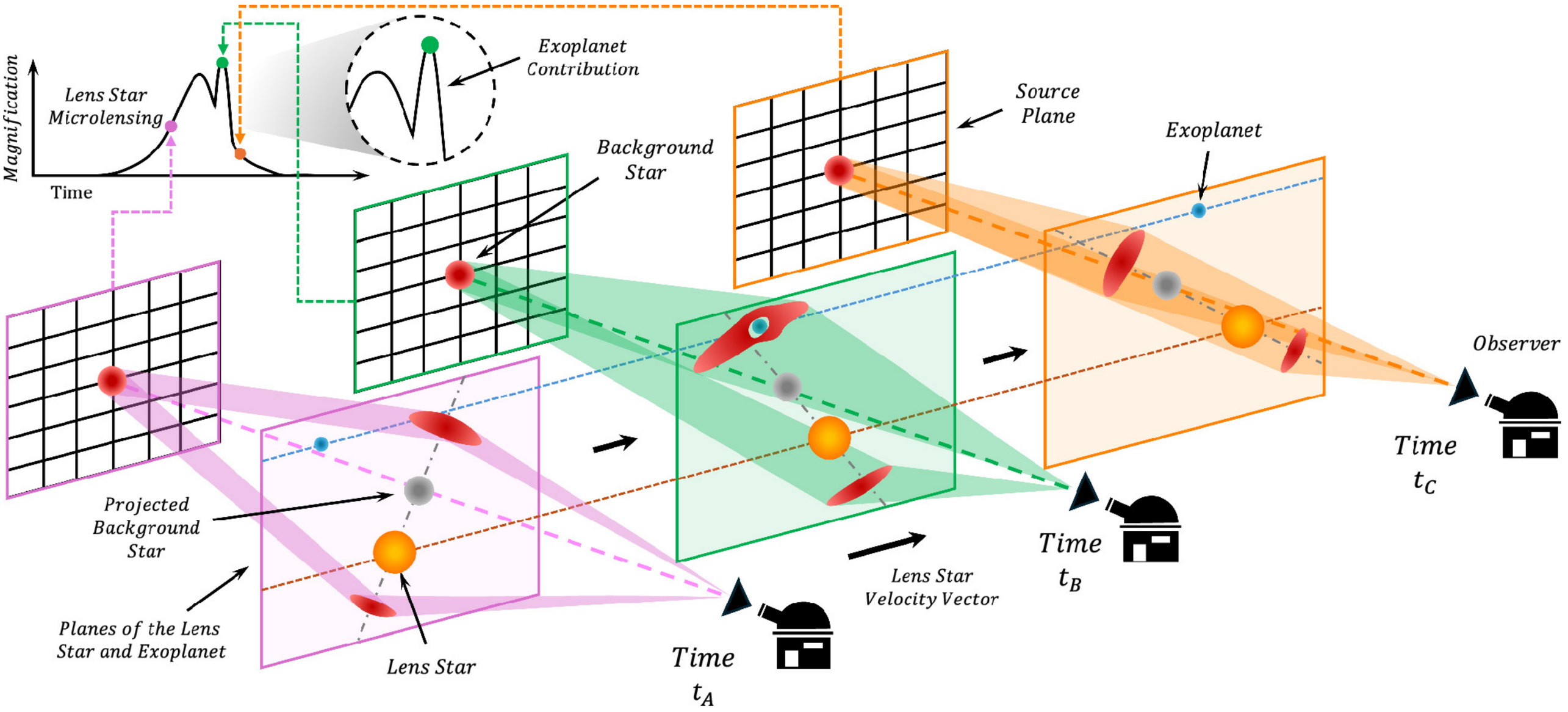
Abstract: key message

- LensNet: machine learning pipeline to work after AlertFinder, replacing manual inspection for vetting. Multibranch RNN architecture, evaluate time-series flux data with diagnosis parameters → classification accuracy ~ 87.5%

Outline of this talk

1. Background: microlensing and similar works
2. KMTNet: survey details, alert finding and preparation of the data
3. Recurrent Neural Networks (RNNs): theory and applications
4. LensNet: architecture, training
5. Results and prospects

1 Introduction: microlensing, exoplanets



$$r_{\mathrm{E}} = D_{\mathrm{d}}\theta_{\mathrm{E}} = \sqrt{\frac{4GM}{c^2}\frac{D_{\mathrm{d}}D_{\mathrm{ds}}}{D_{\mathrm{s}}}} \approx 2.2\,\mathrm{AU}\sqrt{4\times\frac{D_{\mathrm{d}}}{D_{\mathrm{s}}}\left(1-\frac{D_{\mathrm{d}}}{D_{\mathrm{s}}}\right)}\left(\frac{D_{\mathrm{s}}}{8\,\mathrm{kpc}}\right)^{1/2}\left(\frac{M}{0.3M_{\odot}}\right)^{1/2}$$

1 Introduction: similar works

- Real-time alert detection (vs. postseason detection): alert microlensing events as early as possible for follow-up observations, defined as “smoothly increasing in brightness above the baseline level”

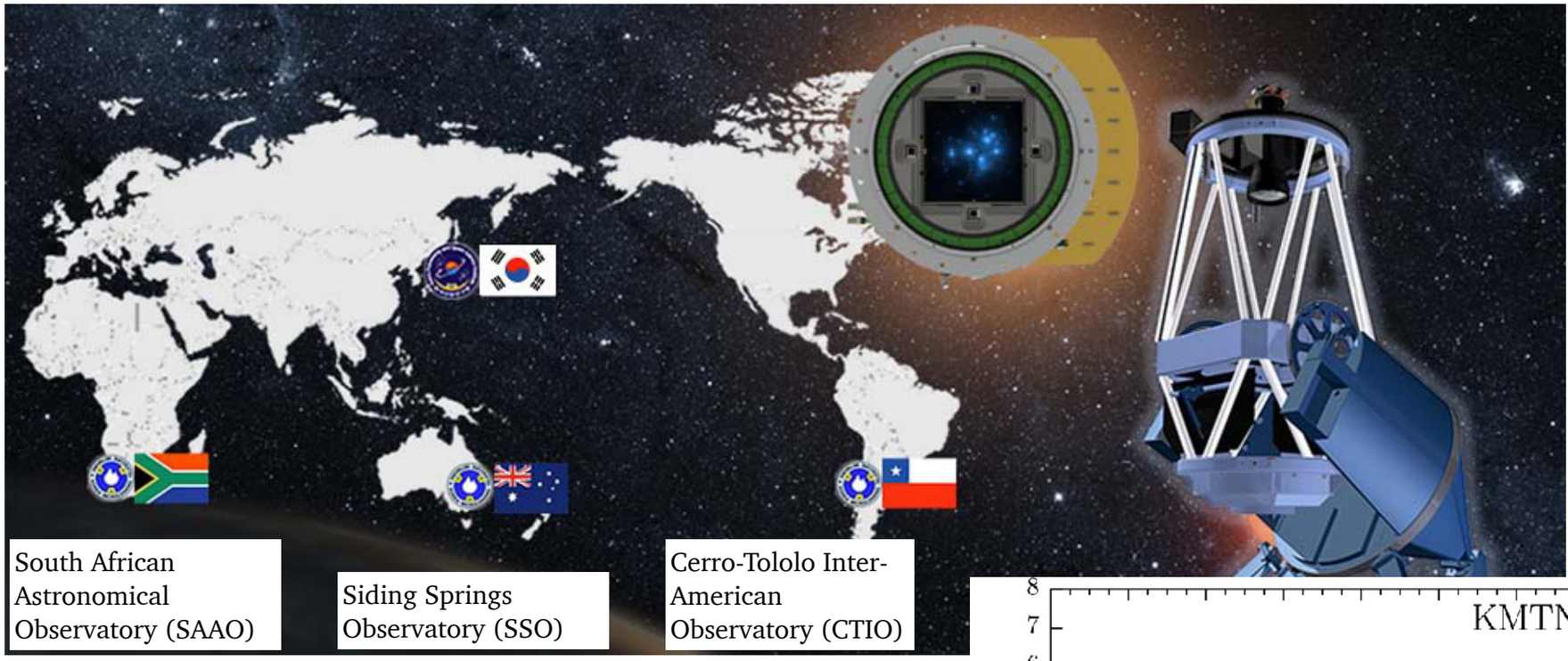
- ML applied to postseason detection:
  - Wyrzykowski et al. (2015): OGLE-III, random forest
  - Chu et al. (2019): UKIRT, random forest
  - Boone (2019): simulated LSST data, gradient-boosted decision trees
  - Mróz (2020): OGLE-III and -IV, convolutional neural networks (CNNs)*
  - Husseinniova et al. (2021): VVV data, decision trees

≈ 98% of single-lens events,
80–85% of binary-lens events.

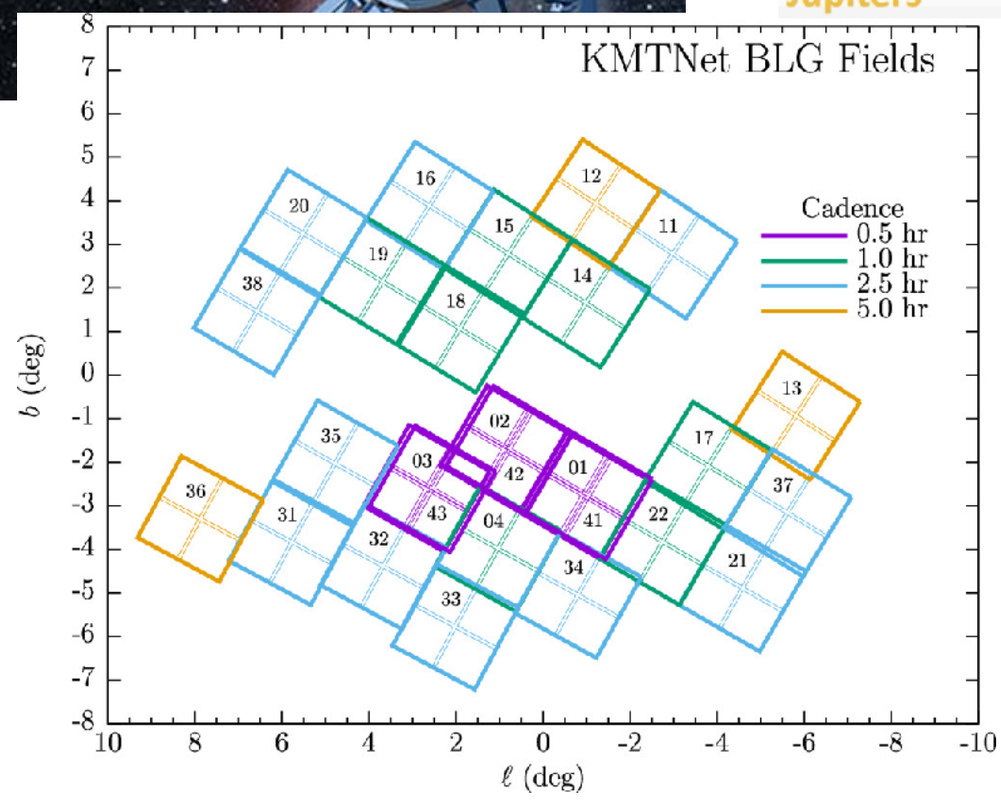
- Random forest applied to real-time detection: Godines et al. (2019, simulated light curves), Gezer et al. (2022, Gaia+ data)

2 KMTNet: survey details

From KMTNet website

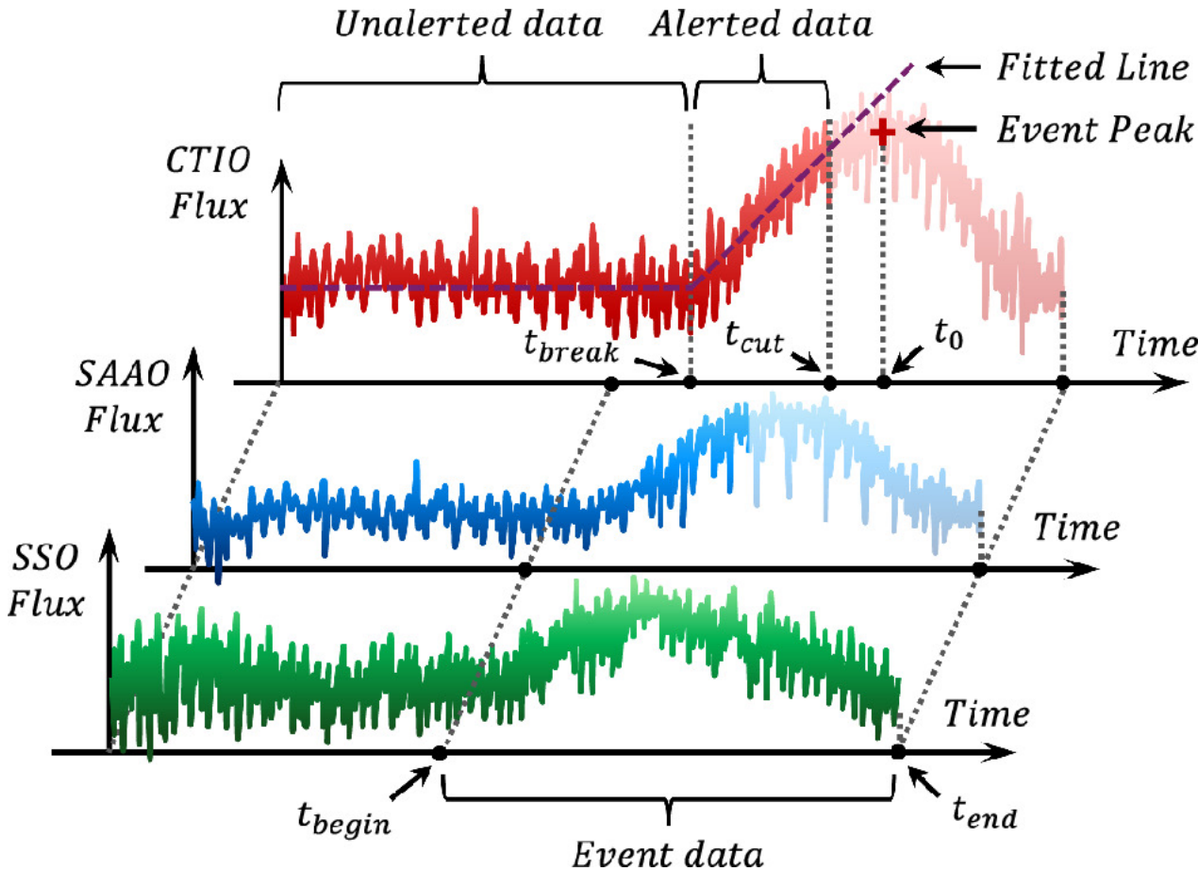


Super-Earths
Neptunes
Saturns
Jupiters

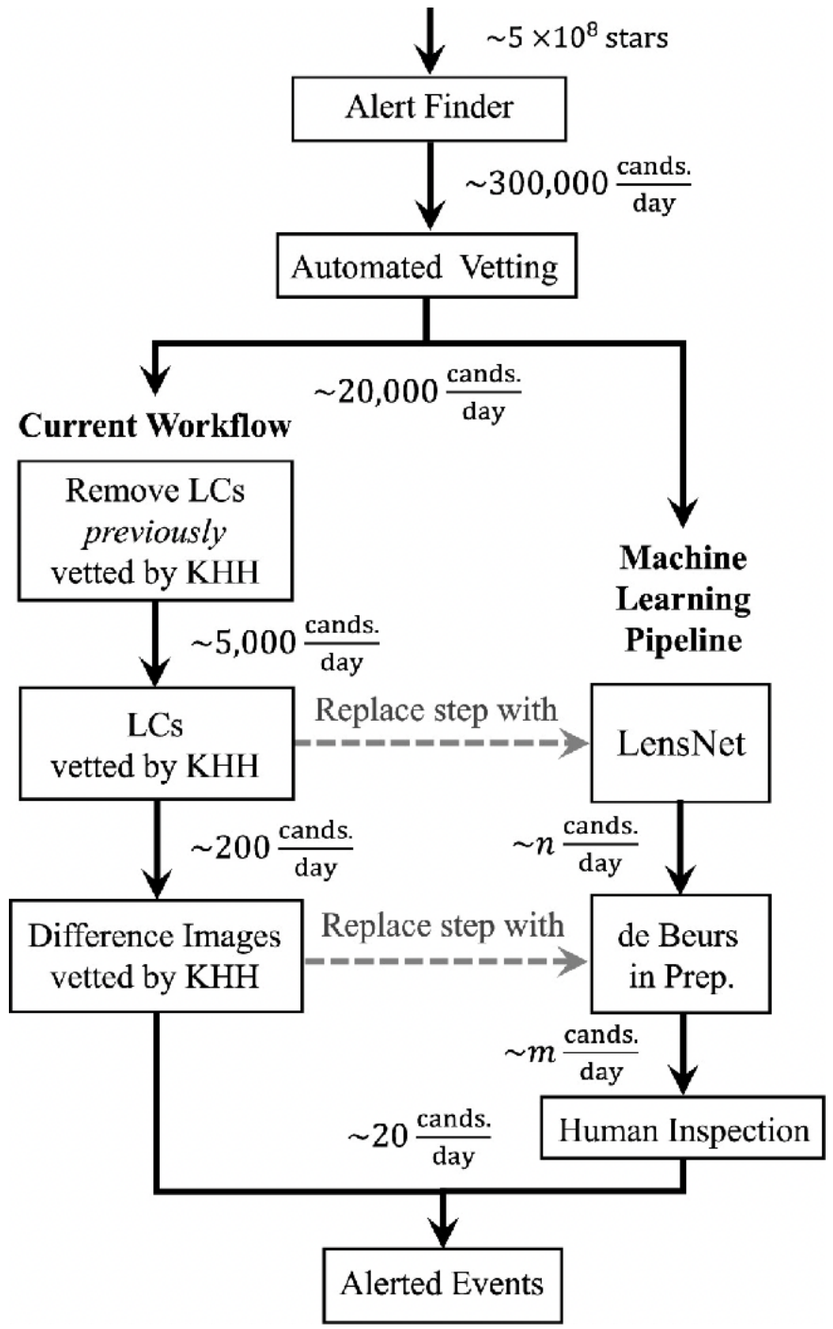


- Three 1.6m telescopes ~ 24h coverage
- BVI filters, field of view of 2x2 degrees
- 500 million stars per night
- Active since 2014 (Kim et al. 2016)

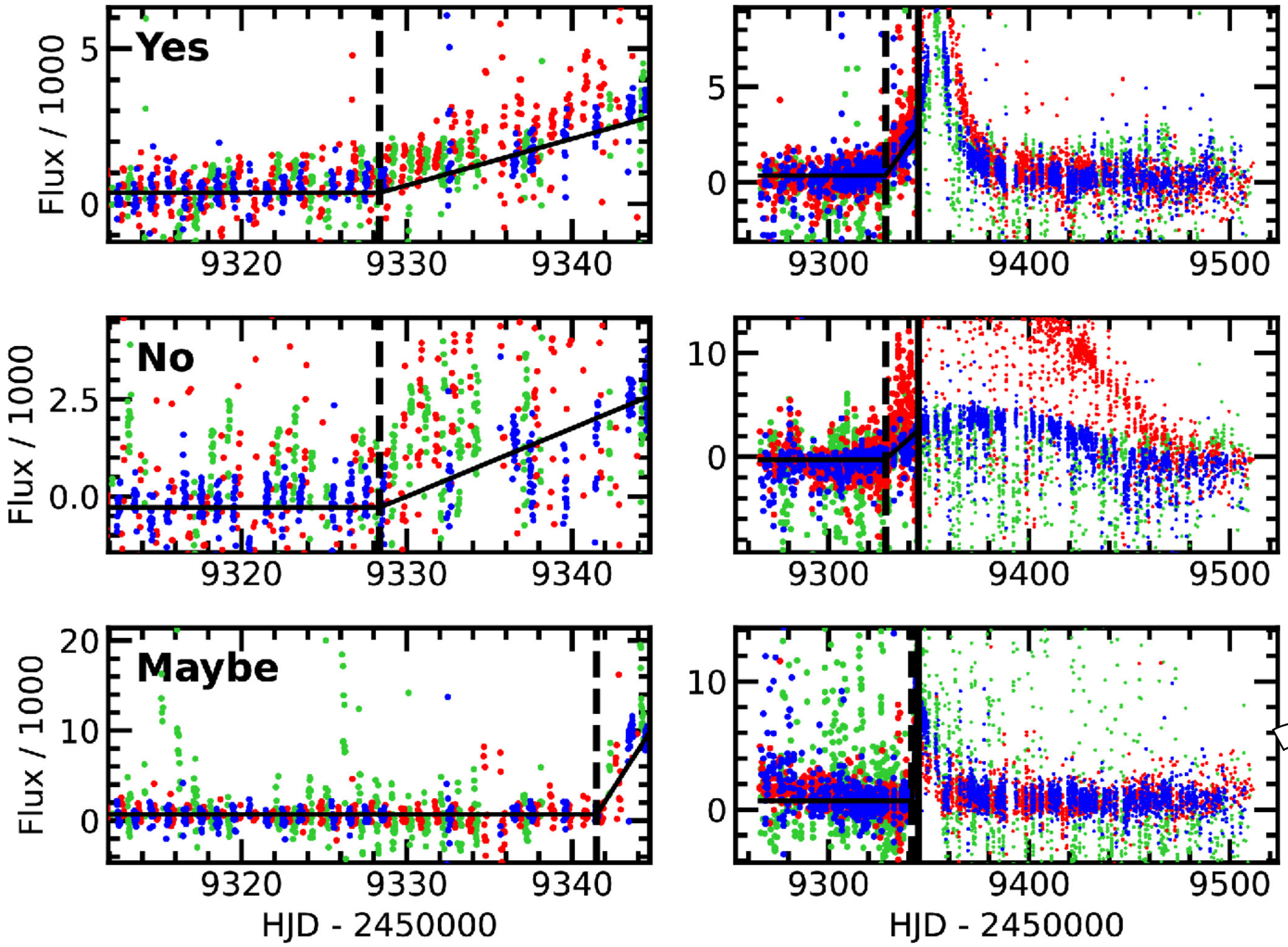
2 KMTNet: AlertFinder and new pipeline

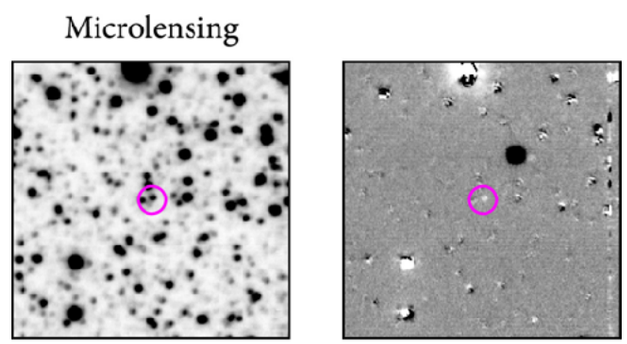


$$F(t) = a_0 + a_1(t - t_{\text{break}})\Theta(t - t_{\text{break}})$$



2 KMTNet: preparation of the data





"Maybe": candidate was identified as a possible event during the light curve review but deemed to be a false positive after examining the images

2 KMTNet: preparation of the input data

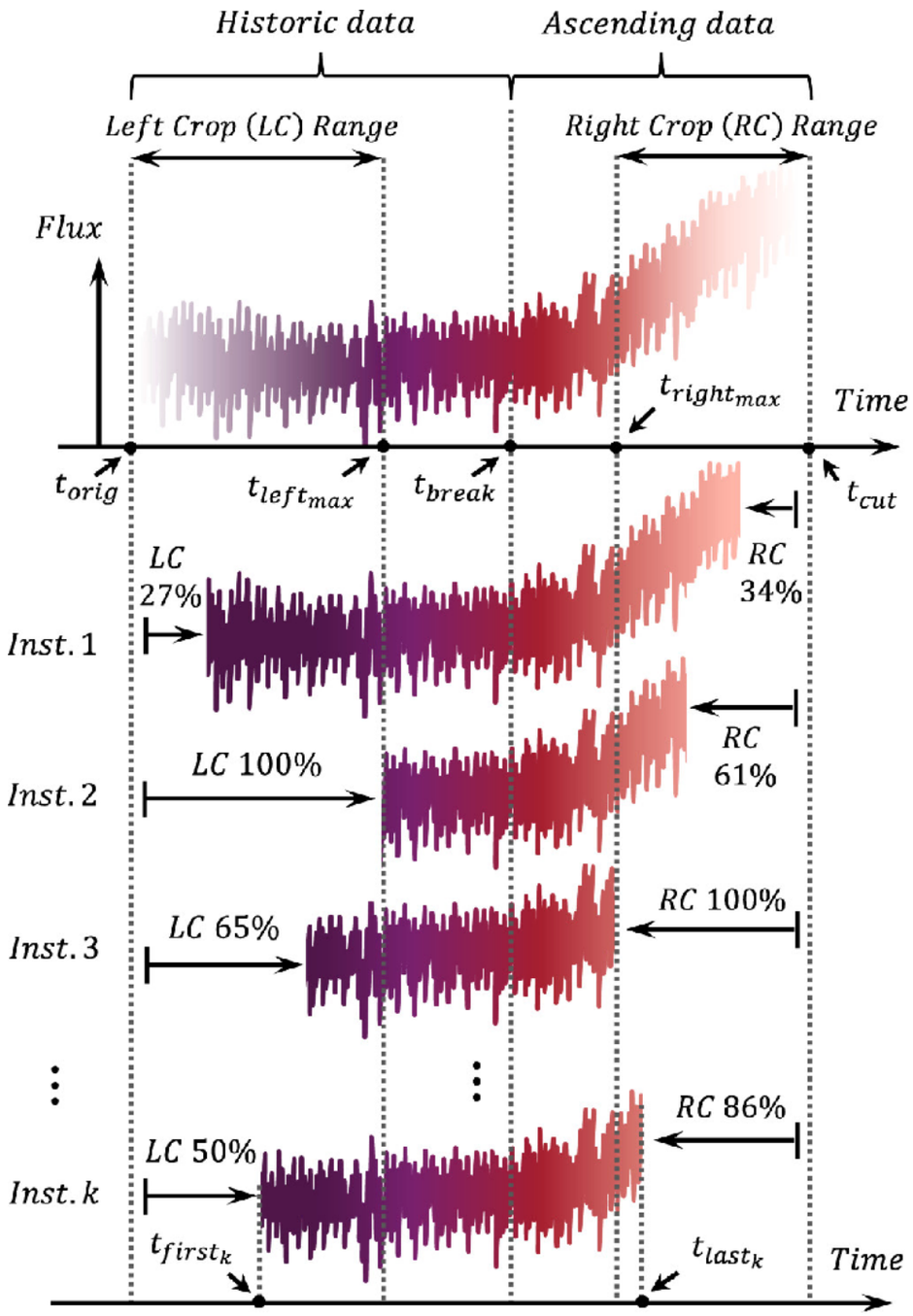


Table 1
Comparison of Nonaugmented and Augmented Instances per Category, Including Totals

Category	Nonaugmented	Augmented
Maybe	1190	7404
No	2038	12,008
Yes	1825	10,902
Total	5053	30,314

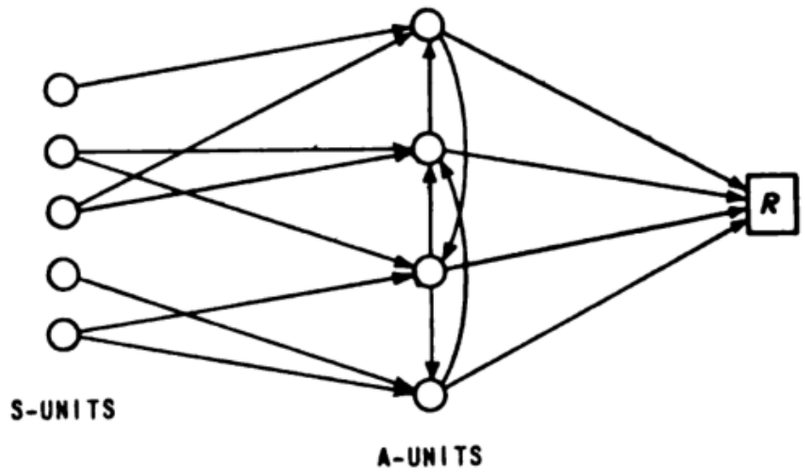
Note. Imbalance difference after the augmentation was intentional to increase the representation of the "maybe" category in the data set, as it is more difficult to classify than the other categories.

Following steps to ensure robustness, good convergence and accuracy:

- Time relativisation: relative to t_{last}
- Outlier removal: data that deviates from the overall trend
- NaN handling: removed
- Fitting the ascending data (AlertFinder)
- Standardisation: normalize the data by subtracting the mean and dividing by the standard deviation
- Padding: append zeros to the beginning of each sequence

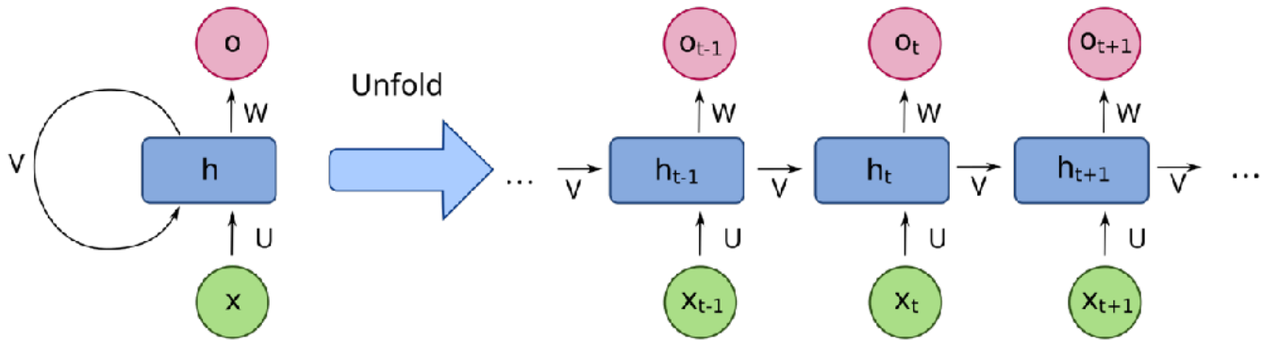
3 Recurrent Neural Networks (RNNs)

- Class of neural networks that allow previous outputs to be used as inputs while having hidden states. Convolutional NNs are feedforward only
- RNNs allow to operate over sequence of vectors (in the input, output or both). It is ideal for time-series data, to model temporal patterns (e.g. correlated noise)
- CNNs for spatial information (e.g. images), RNNs for temporal and sequential data (e.g. text, videos, NLP, language translation, image captioning)



By Frank Rosenblatt, 1961

From Wikipedia (fdeloche)



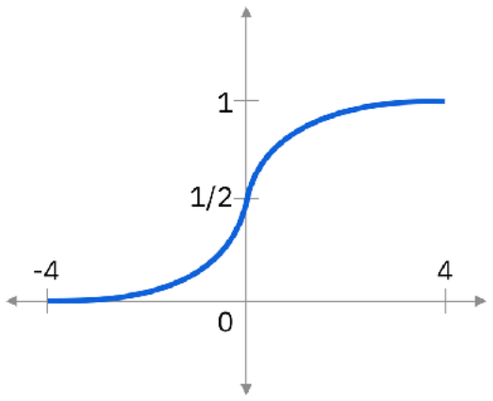
$$y_t = \alpha\left(W_x x_t + W_y y_{t-1} + b\right)$$

3 Recurrent Neural Networks (RNNs)

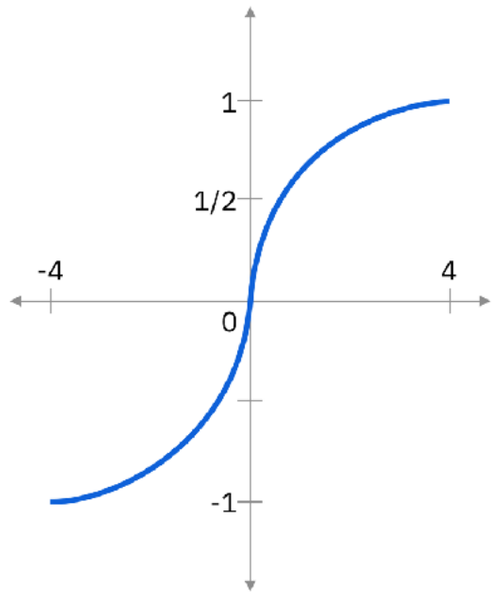
- RNNs share the same weight within each layer, the weights are adjusted through backpropagation and gradient descent. Cost ↔ gradients ↔ update weights
- Common activation functions: sigmoid function, hyperbolic tangent* (Tanh), Rectified Linear Unit (ReLU)

From IBM website

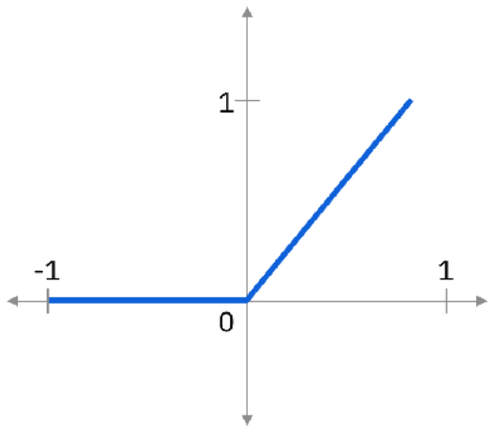
$$g(x)=\frac{1}{1+e^{-x}}$$



$$g(x)=\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$$

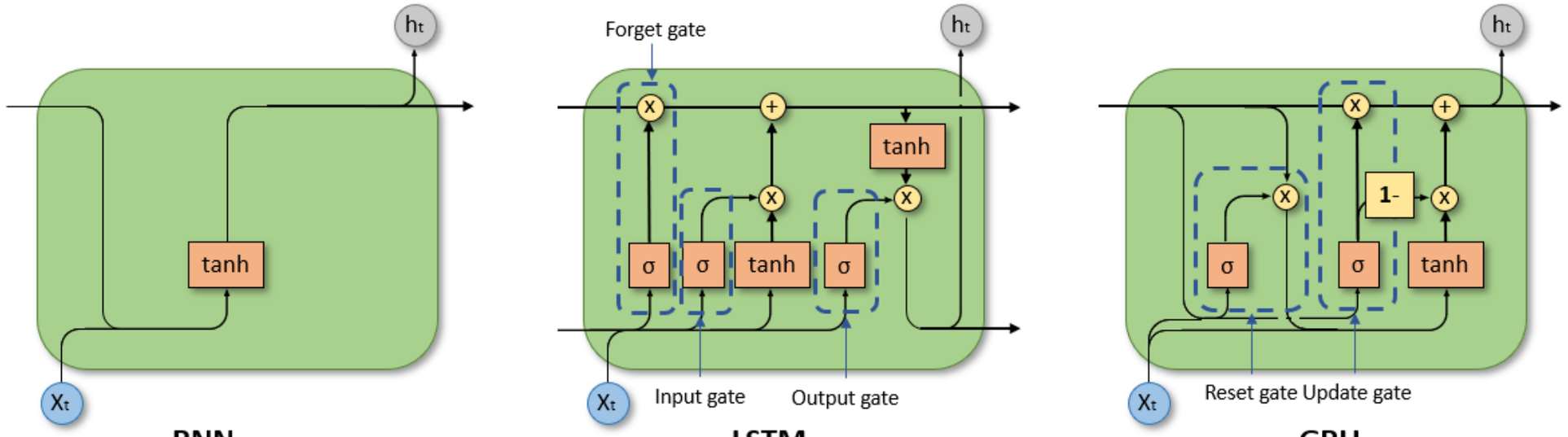


$$g(x)=\max(0,x)$$



3 Recurrent Neural Networks (RNNs)

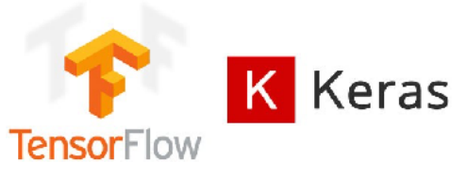
- RNNs share the same weight within each layer, the weights are adjusted through backpropagation and gradient descent. Cost ↔ gradients ↔ update weights
- Common activation functions: sigmoid function, hyperbolic tangent* (Tanh), Rectified Linear Unit (ReLU)
- Problems: unstable gradients (vanishing vs. exploding), long-term memory doesn't work well → types: bidirectional RNNs, long short-term memory* (LSTM), gated recurrent units (GRUs), encoder-decoder



From "Towards Data Science"

3 Recurrent Neural Networks (RNNs)

- RNNs share the same weight within each layer, the weights are adjusted through backpropagation and gradient descent. Cost ↔ gradients ↔ update weights
- Common activation functions: sigmoid function, hyperbolic tangent* (Tanh), Rectified Linear Unit (ReLU)
- Problems: unstable gradients (vanishing vs. exploding), long-term memory doesn't work well → types: bidirectional RNNs, long short-term memory* (LSTM), gated recurrent units (GRUs), encoder-decoder
- Keras is a popular implementation, integrated into TensorFlow Python library
- IBM website: *The use of RNNs is declining but it is not obsolete. Transformer models such as BERT (pre-trained bidirectional transformer) and GPT (generative pre-trained transformer) can capture long-term dependencies more effectively, are easier to parallelise and perform NLP better.*



3 Recurrent Neural Networks (RNNs)

From Medium website
(Carolina Bento)

1 2020MNRAS.491.4277M 2020/01 cited: 161
SuperNNova: an open-source framework for Bayesian, neural network-based supernova classification
Möller, A.; de Boissière, T.

2 2019PASP..131k8002M 2019/11 cited: 145
RAPID: Early Classification of Explosive Transients Using Deep Learning
Muthukrishna, Daniel; Narayan, Gautham; Mandel, Kaisey S. *and 2 more*

3 2020GeoRL..4785976M 2020/01 cited: 137
A Machine-Learning Approach for Earthquake Magnitude Estimation
Mousavi, S. Mostafa; Beroza, Gregory C.

4 2017ApJ...837L..28C 2017/03 cited: 104
Deep Recurrent Neural Networks for Supernovae Classification
Charnock, Tom; Moss, Adam

5 2018NatAs...2..151N 2018/11 cited: 97
A recurrent neural network for classification of unevenly sampled variable stars
Naul, Brett; Bloom, Joshua S.; Pérez, Fernando *and 1 more*

6 2019GeoRL..46.3643L 2019/04 cited: 79
Deep Learning Models Augment Analyst Decisions for Event Discrimination
Linville, Lisa; Pankow, Kristine; Draelos, Timothy

7 2020JCAP...03..008E 2020/03 cited: 70
A deep learning approach to cosmological dark energy models
Escamilla-Rivera, Celia; Carvajal Quintero, Maryi A.; Capozziello, Salvatore

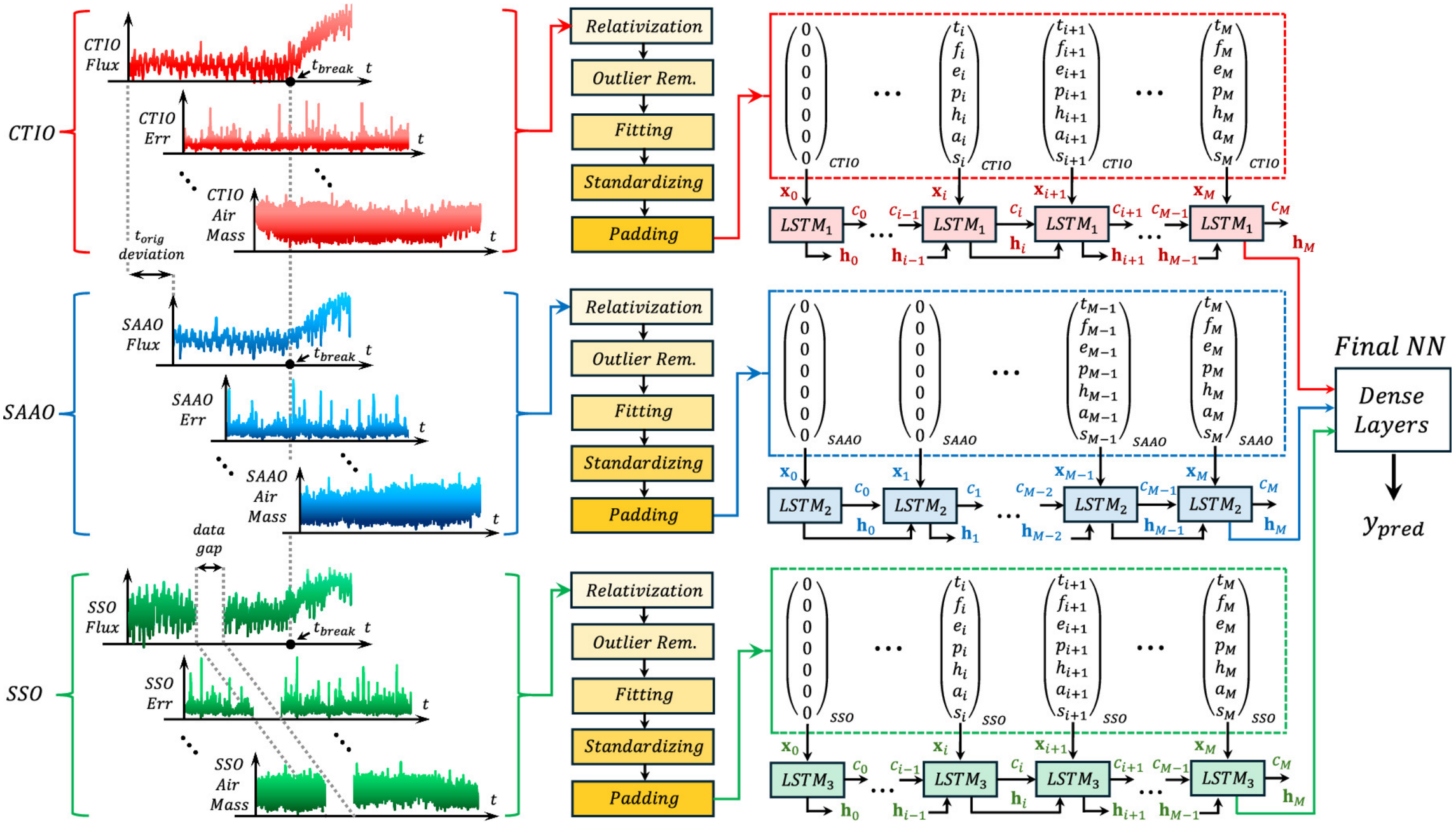
8 2019APh...105...44S 2019/02 cited: 65
Application of deep learning methods to analysis of imaging atmospheric Cherenkov telescopes data
Shilon, I.; Kraus, M.; Büchele, M. *and 7 more*

ADS search for “RNN” or “Recurrent Neural Network” in the abstract. Filtered by refereed papers, sorted by citation count

1 model_predictions.py

```python
import os
import tensorflow as tf

# Loading Training and Test Datasets
train_dir = os.path.join('', '../datasets/train')

train_dataset = tf.keras.utils.text_dataset_from_directory(
    train_dir, label_mode = 'int', labels = 'inferred', follow_links = True
)

test_dir = os.path.join('', '../datasets/test')

test_dataset = tf.keras.utils.text_dataset_from_directory(
    test_dir, label_mode = 'int', labels = 'inferred', follow_links = True
)

# Vectorize training dataset
VOCAB_SIZE = 5000
encoder = tf.keras.layers.TextVectorization(max_tokens=VOCAB_SIZE)
encoder.adapt(train_dataset.map(lambda text, label: text))

# Building the Recurrent Neural Network
# using GRU cells and Hyperbolic tangent as activation function
cell = tf.keras.layers.GRUCell(30, recurrent_activation='tanh')
model = tf.keras.Sequential([
    encoder,
    tf.keras.layers.Embedding(
        input_dim=len(encoder.get_vocabulary()),
        output_dim=64,
        # Use masking to handle the variable sequence lengths
        mask_zero=True),
    tf.keras.layers.Bidirectional(tf.keras.layers.RNN(cell)),
    tf.keras.layers.Dense(60, activation='tanh'),
    tf.keras.layers.Dense(1)
])

# Compile model and use the algorithm Adam as optimization function
model.compile(loss=tf.keras.losses.BinaryCrossentropy(from_logits=True),
              optimizer=tf.keras.optimizers.Adam(1e-2),
              metrics=['accuracy'])

# Fitting the model
history = model.fit(train_dataset, epochs=10, validation_data=test_dataset, validation_steps=10)

# Model Evaluation
test_loss, test_acc = model.evaluate(test_dataset)

print('Test Loss:', test_loss)
print('Test Accuracy:', test_acc)
```

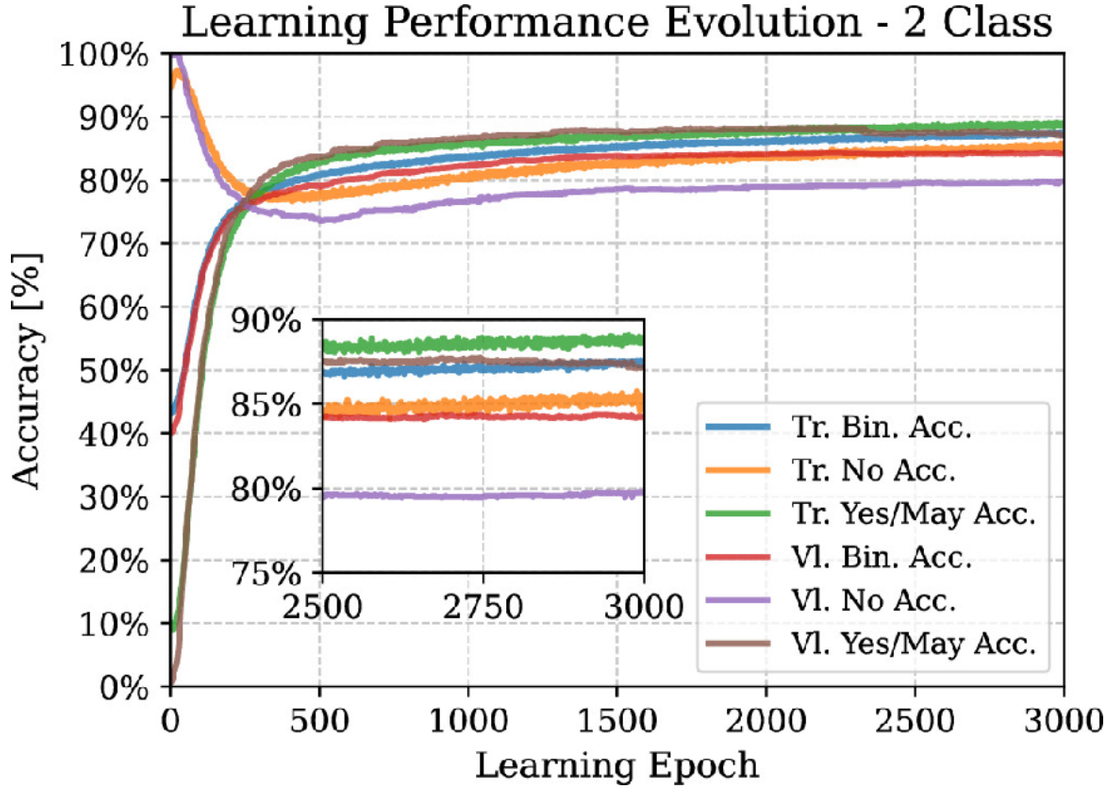
4 LensNet: RNN architecture

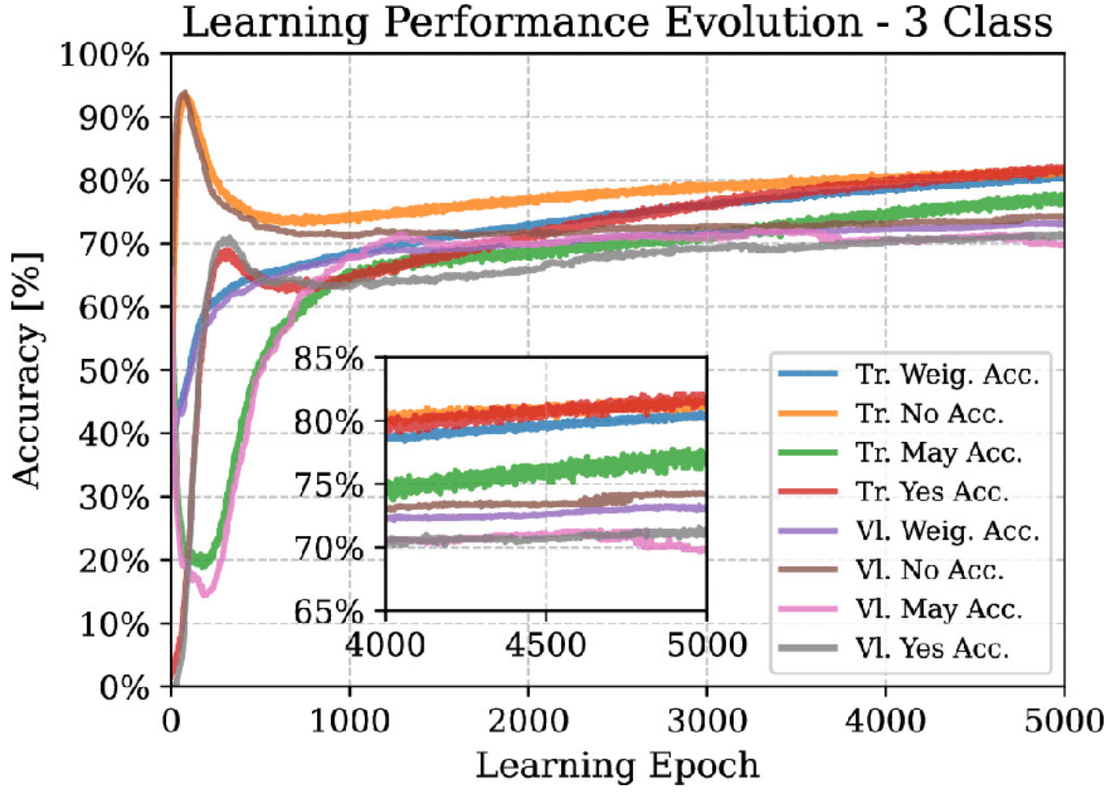


4 LensNet: RNN architecture

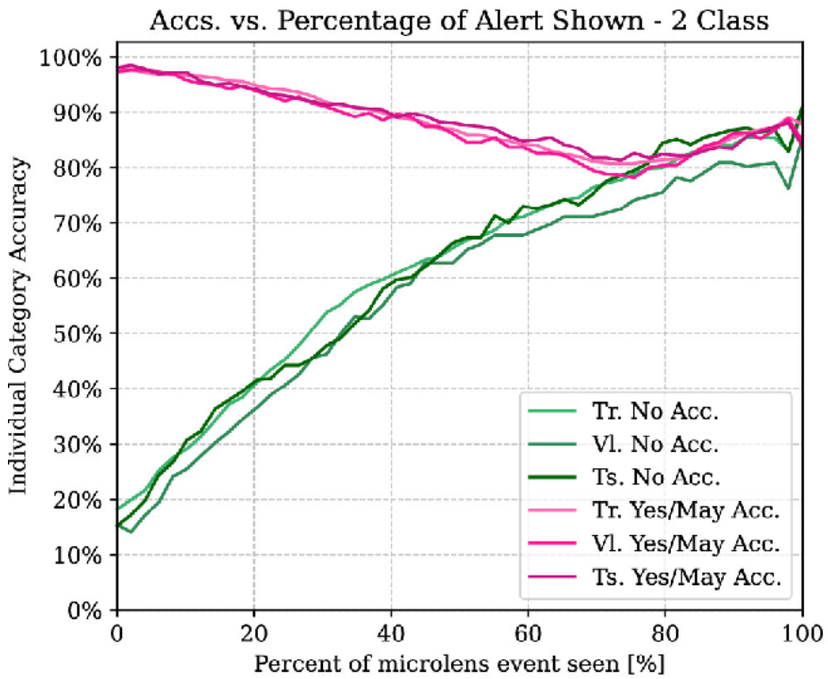
The authors applied in this work:

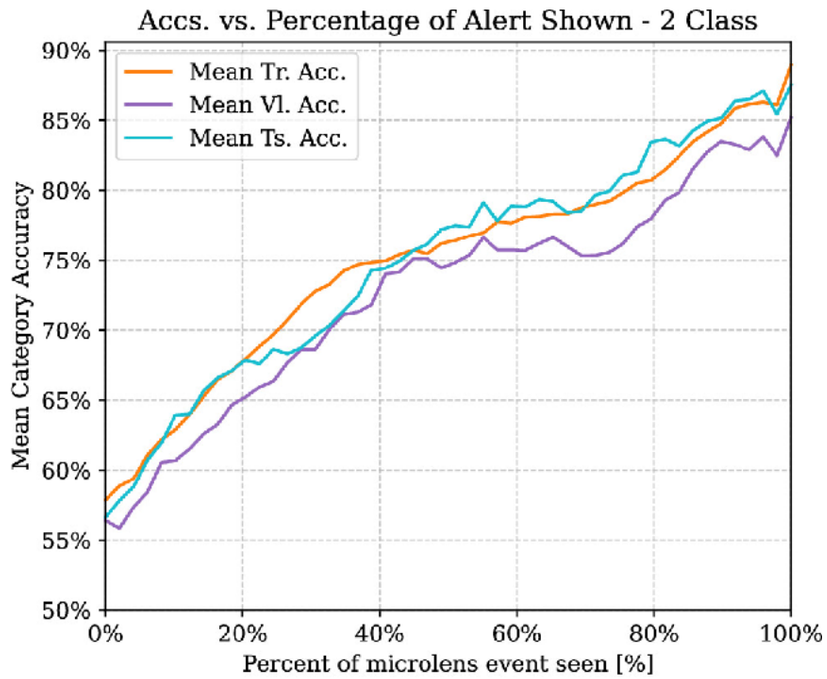
- TensorFlow (Abadi et al. 2015)
- 4 NVIDIA A100s GPUs, 128 CPU cores, 100 GB of RAM
- Adam optimiser (Kingma 2014)
- Different loss functions for each type of classification (Mao et al. 2023):
  - Binary Cross-Entropy (Yes+Maybe vs No)
  - Categorical Cross-Entropy (Yes vs Maybe vs No)

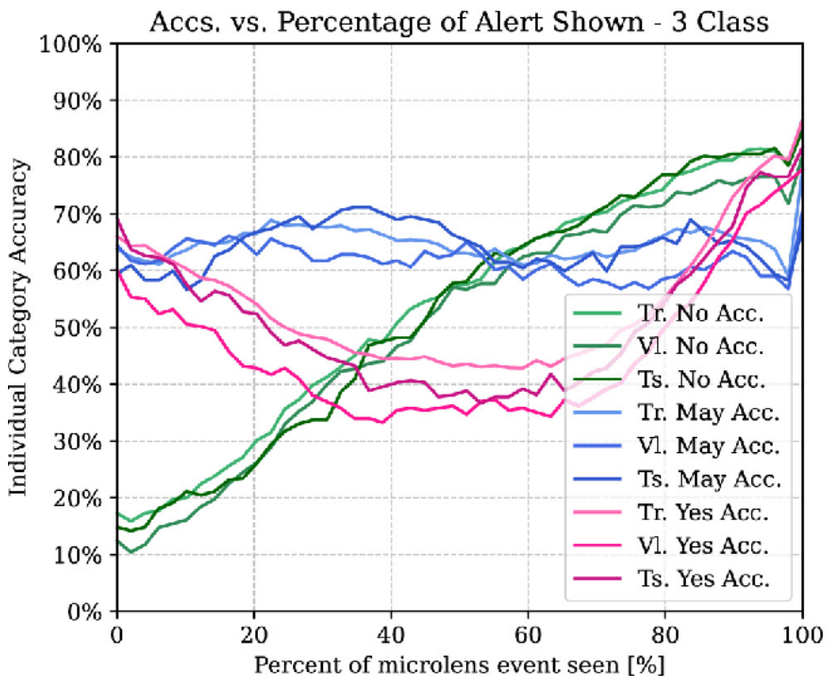


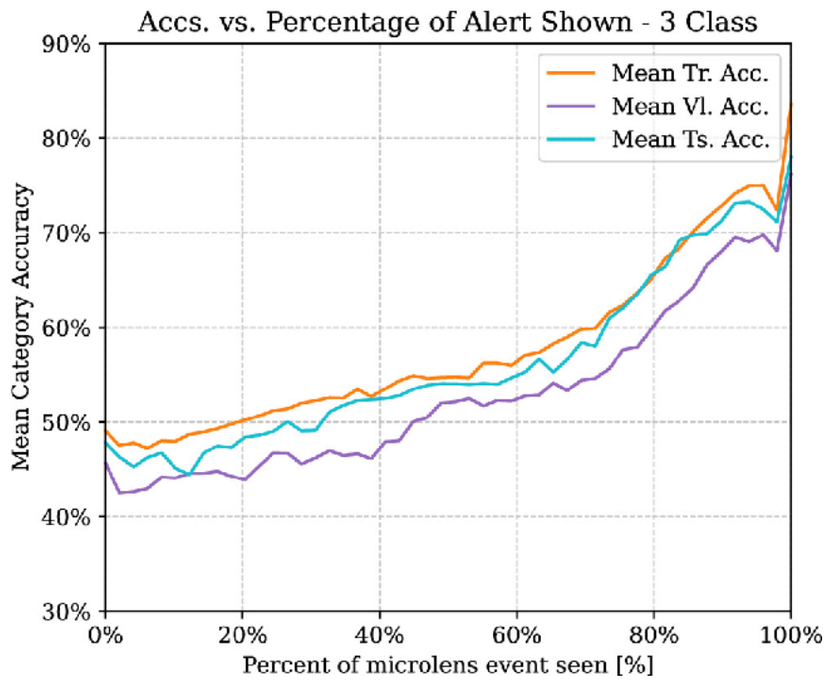


5 Results: category accuracy (2 vs. 3 classes)

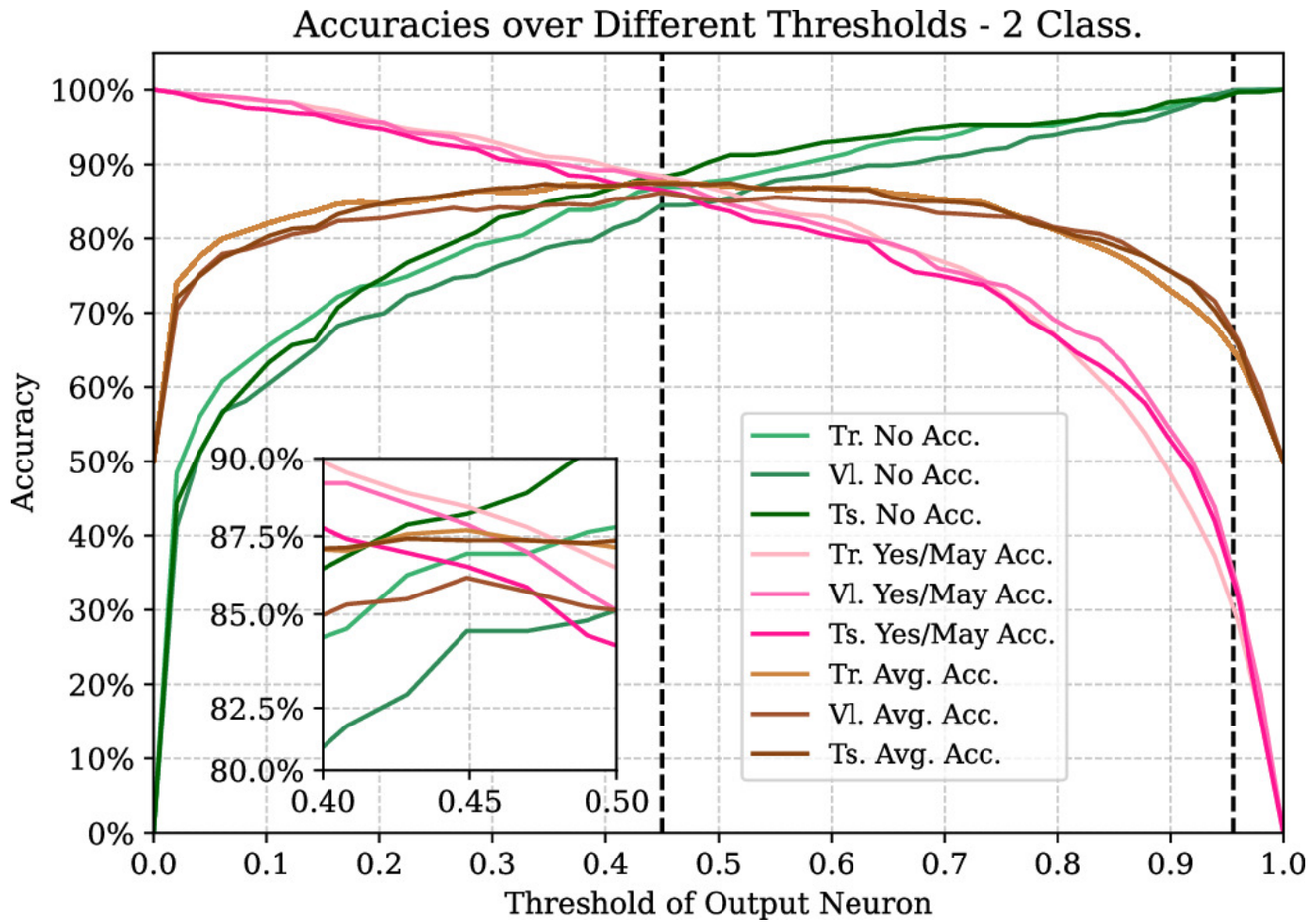








5 Results: threshold and prospects



- LensNet will possibly be integrated to work with AlertFinder, avoiding manual vetting
- Different thresholds used: 0.45 (87.5%) vs. ~1.0 (99.7%)
- Second stage to check difference imaging (de Beurs in prep.), more memory allocation