Hluboke uceni detekce malwaru na EXE souborech bez predzpracovani

Marek Krčál, Avast fellow at Institute of Computer Science

Joint work with Ondřej Švec, Martin Bálek and Otakar Jašek

Detection of malicious executables

- **static analysis**: header, entropy/function call profiles, strings, even n-grams as features for ML

Detection of malicious executables

- **static analysis**: header, entropy/function call profiles, strings, even *n*-grams as features for ML

(dynamic analysis (simulating the execution): API calls, their parameters, null terminated objects, memory allocation patterns)

Detection of malicious executables

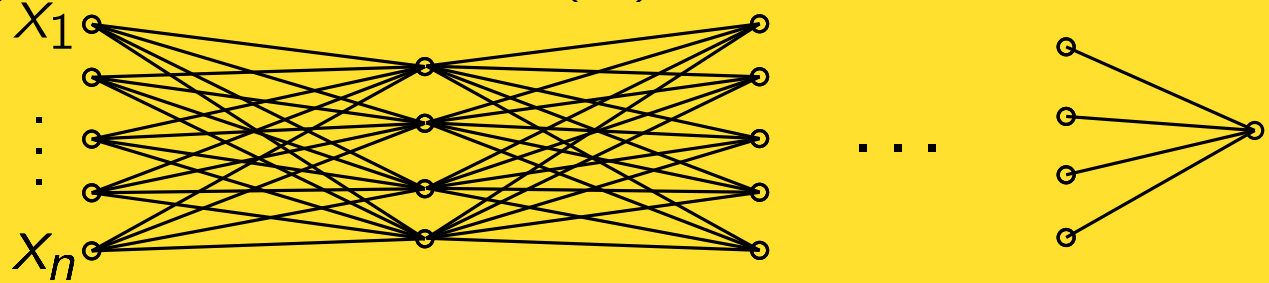
- **static analysis**: header, entropy/function call profiles, strings, even *n*-grams as features for ML

(dynamic analysis (simulating the execution): API calls, their parameters, null terminated objects, memory allocation patterns)

- **avoiding false positives**
  - false malware detections ruin AV bussiness
  - much less malware in the wild (than needed to learn a classifier)

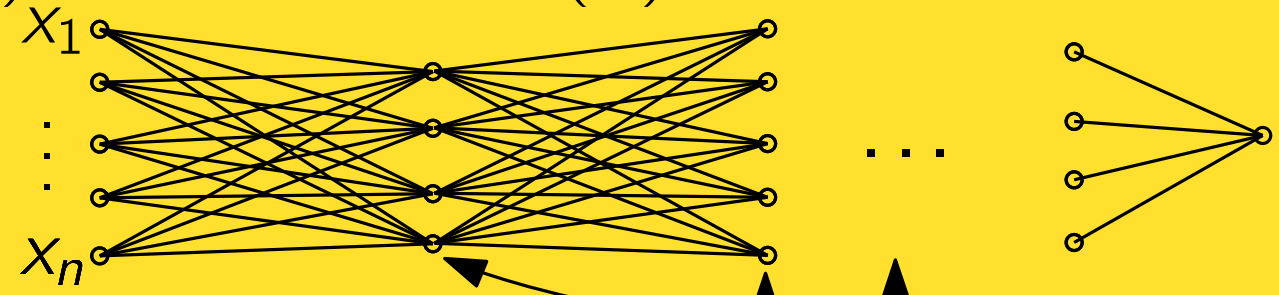
Deep Learning

$$f(\boldsymbol{x}) = f^{(k)} \dots f^{(2)} f^{(1)}(\boldsymbol{x}), \ \boldsymbol{\Theta}^{(k)}, \dots, \boldsymbol{\Theta}^{(2)}, \boldsymbol{\Theta}^{(1)}$$



Deep Learning

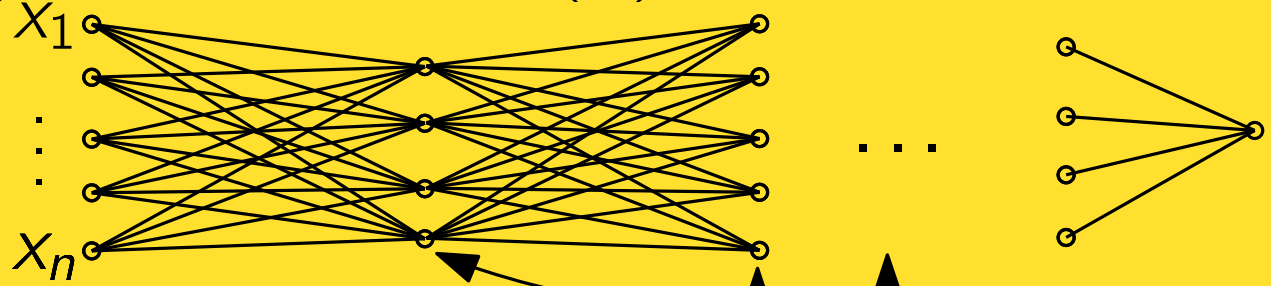
$$f(\boldsymbol{x}) = f^{(k)} \dots f^{(2)} f^{(1)}(\boldsymbol{x}), \ \boldsymbol{\Theta}^{(k)}, \dots, \boldsymbol{\Theta}^{(2)}, \boldsymbol{\Theta}^{(1)}$$



- **end-to-end, feature extraction**

Deep Learning

$$f(\boldsymbol{x}) = f^{(k)} \dots f^{(2)} f^{(1)}(\boldsymbol{x}), \ \boldsymbol{\Theta}^{(k)}, \dots, \boldsymbol{\Theta}^{(2)}, \boldsymbol{\Theta}^{(1)}$$

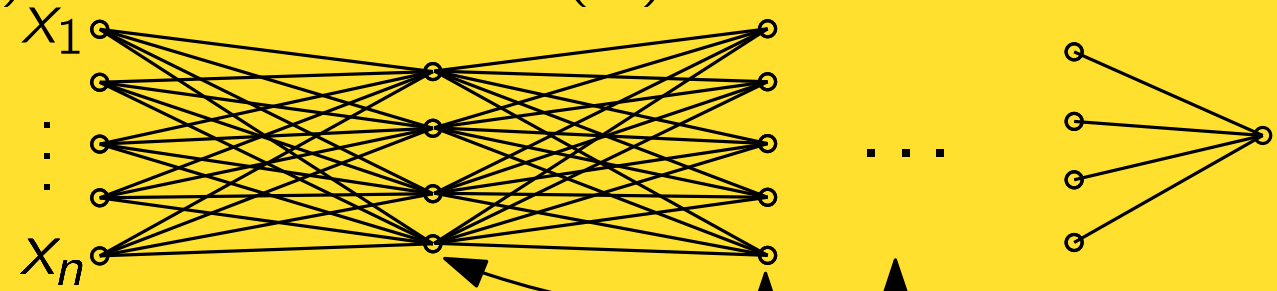


- **end-to-end, feature extraction**

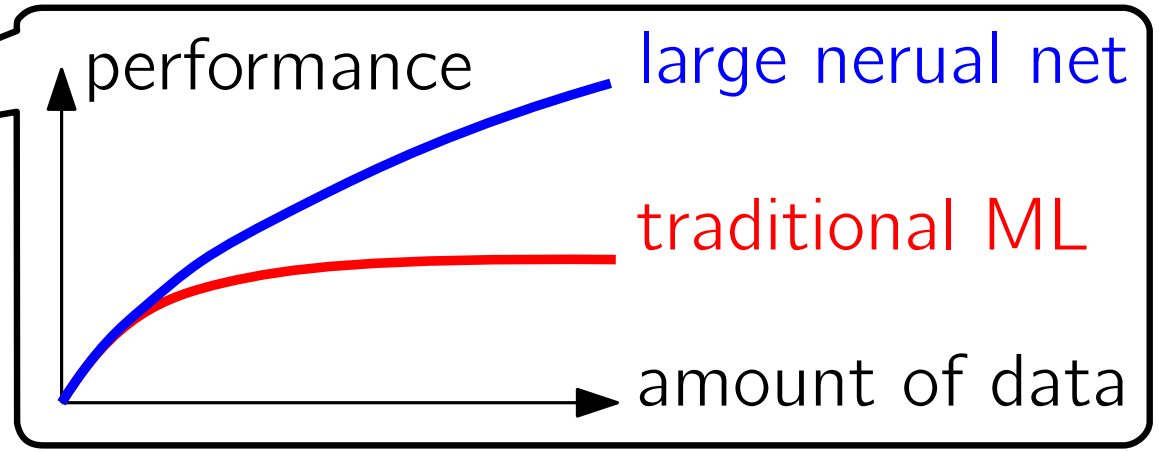
- **data-hungry**

Deep Learning

$$f(\boldsymbol{x}) = f^{(k)} \dots f^{(2)} f^{(1)}(\boldsymbol{x}), \quad \boldsymbol{\Theta}^{(k)}, \dots, \boldsymbol{\Theta}^{(2)}, \boldsymbol{\Theta}^{(1)}$$

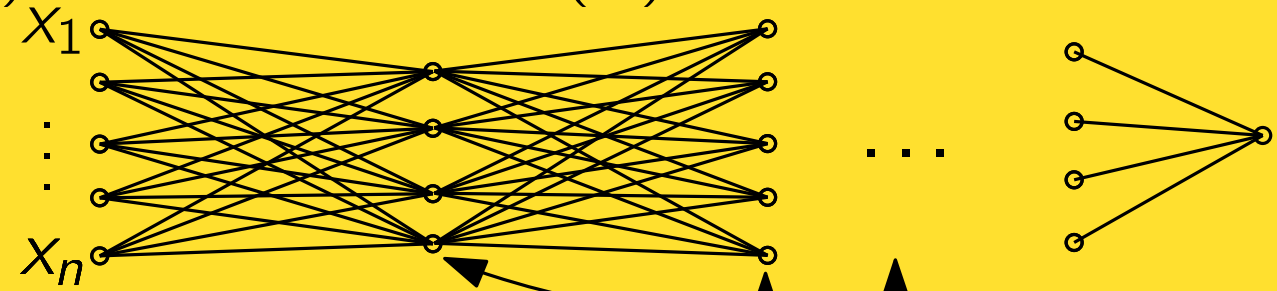


- **end-to-end, feature extraction**
- **data-hungry**



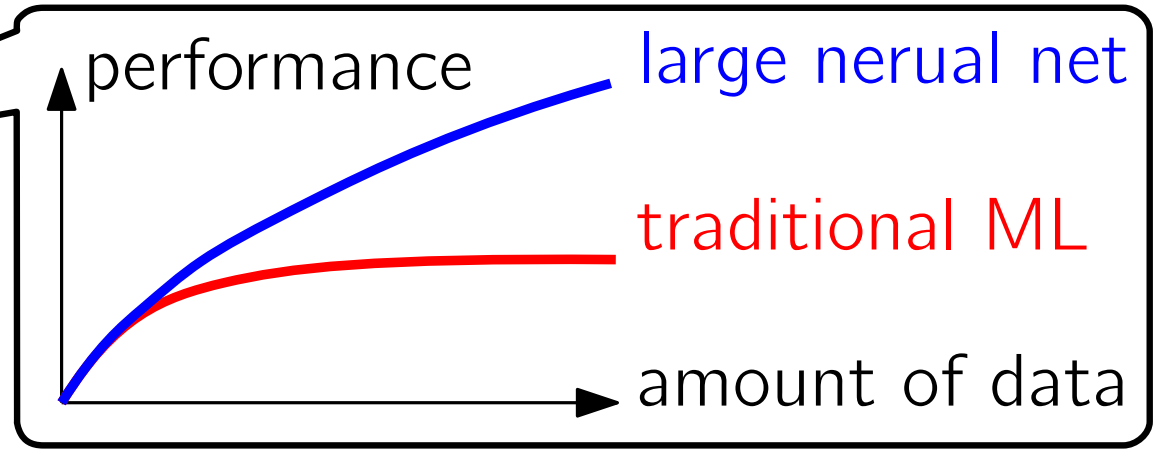
Deep Learning

$$f(\boldsymbol{x}) = f^{(k)} \dots f^{(2)} f^{(1)}(\boldsymbol{x}), \quad \boldsymbol{\Theta}^{(k)}, \dots, \boldsymbol{\Theta}^{(2)}, \boldsymbol{\Theta}^{(1)}$$



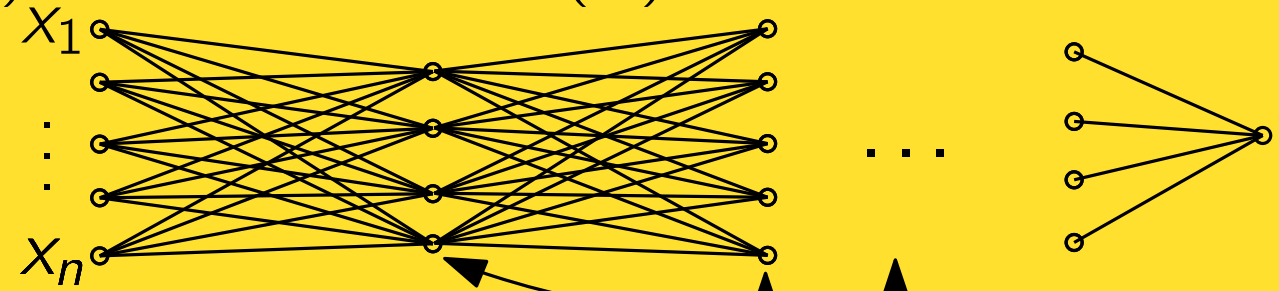
- **end-to-end, feature extraction**

- **data-hungry**
  training from HDD

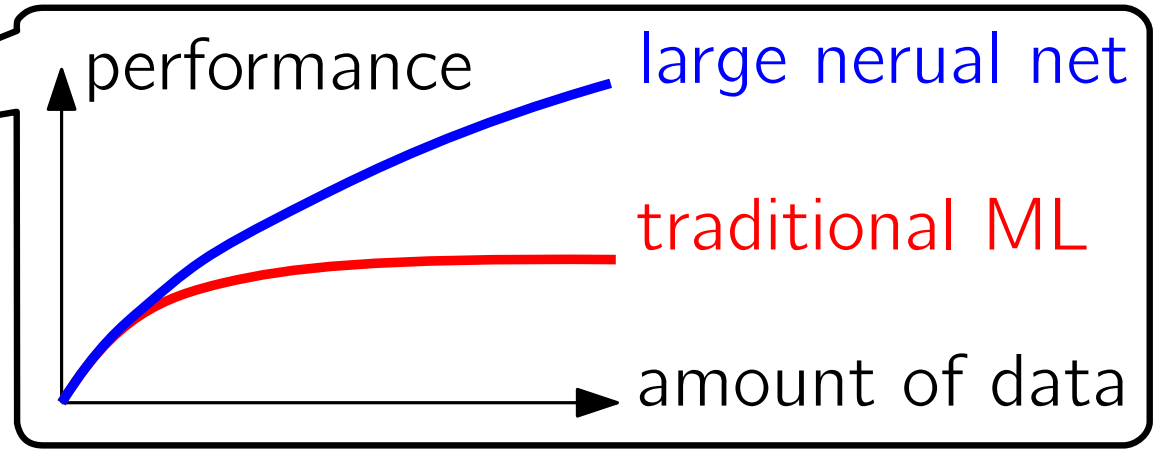


Deep Learning

$$f(\boldsymbol{x}) = f^{(k)} \dots f^{(2)} f^{(1)}(\boldsymbol{x}), \quad \boldsymbol{\Theta}^{(k)}, \dots, \boldsymbol{\Theta}^{(2)}, \boldsymbol{\Theta}^{(1)}$$

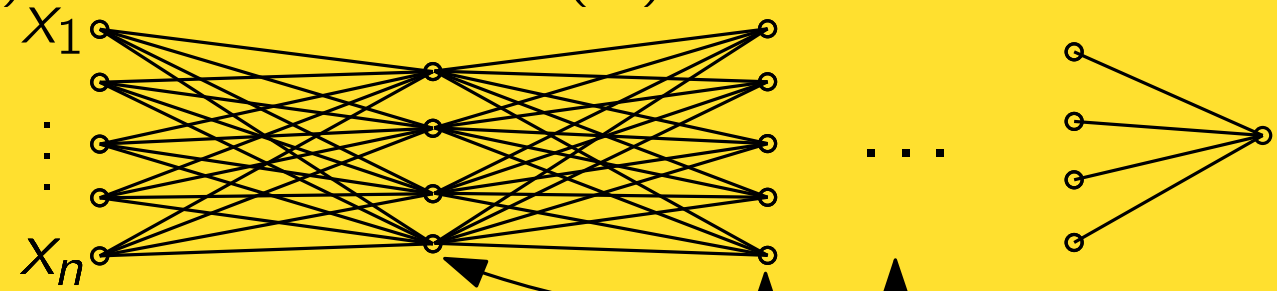


- **end-to-end, feature extraction**
- **data-hungry**
  training from HDD
- **variable-length input**



Deep Learning

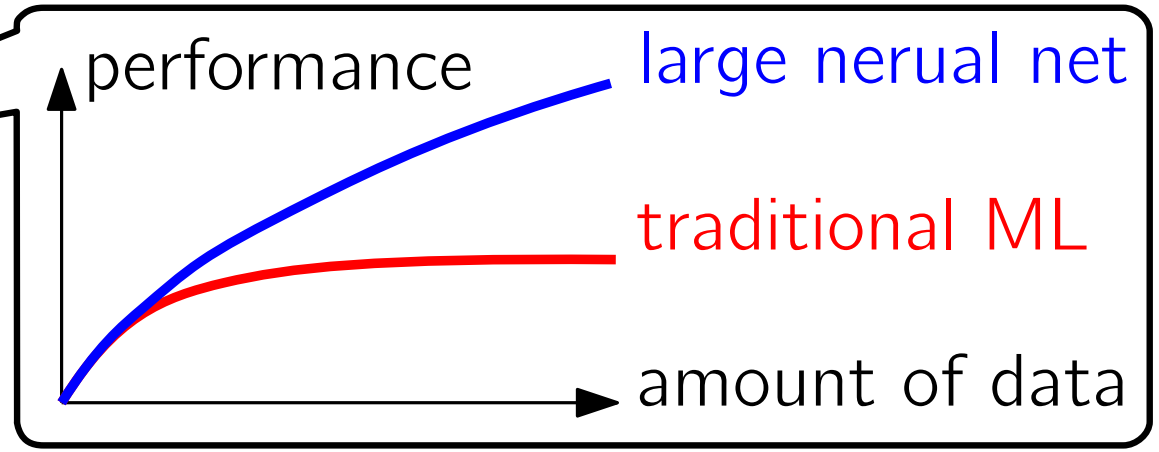
$$f(\boldsymbol{x}) = f^{(k)} \dots f^{(2)} f^{(1)}(\boldsymbol{x}), \quad \boldsymbol{\Theta}^{(k)}, \dots, \boldsymbol{\Theta}^{(2)}, \boldsymbol{\Theta}^{(1)}$$



- **end-to-end, feature extraction**

- **data-hungry**

  training from HDD

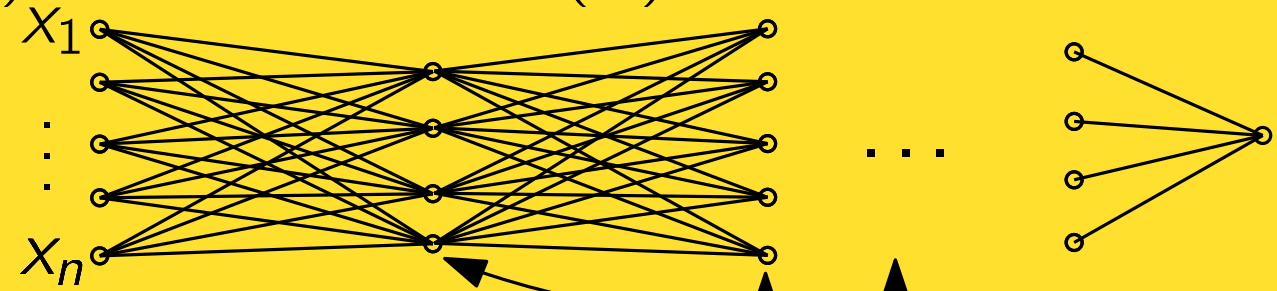


- **variable-length input**

  sequences by recurrent nets mostly

Deep Learning

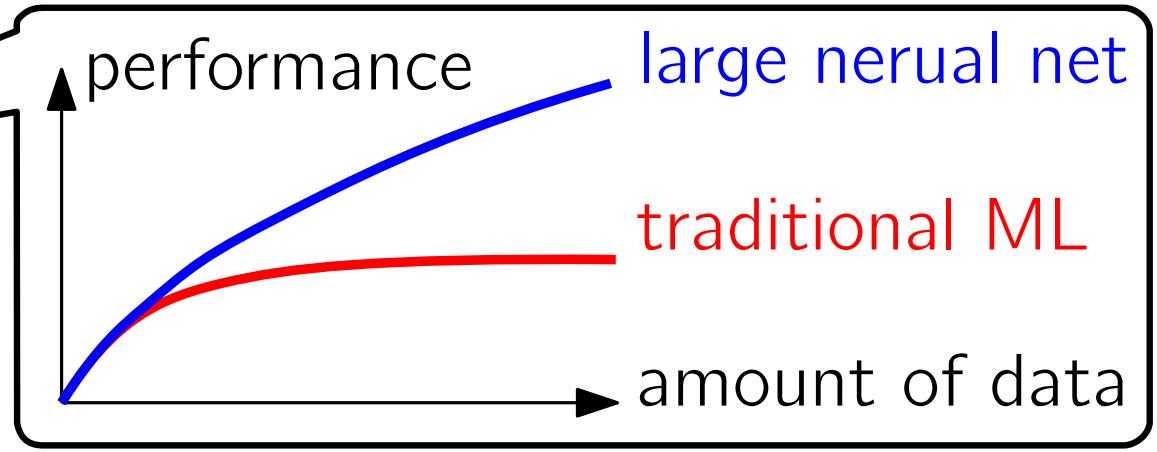
$$f(\boldsymbol{x}) = f^{(k)} \dots f^{(2)} f^{(1)}(\boldsymbol{x}), \quad \boldsymbol{\Theta}^{(k)}, \dots, \boldsymbol{\Theta}^{(2)}, \boldsymbol{\Theta}^{(1)}$$



- **end-to-end, feature extraction**

- **data-hungry**

  training from HDD



- **variable-length input**

  sequences by recurrent nets mostly

  too long sequences: 1D convolutions followed by max/avg

Convolutional Nets

Convolutional Nets

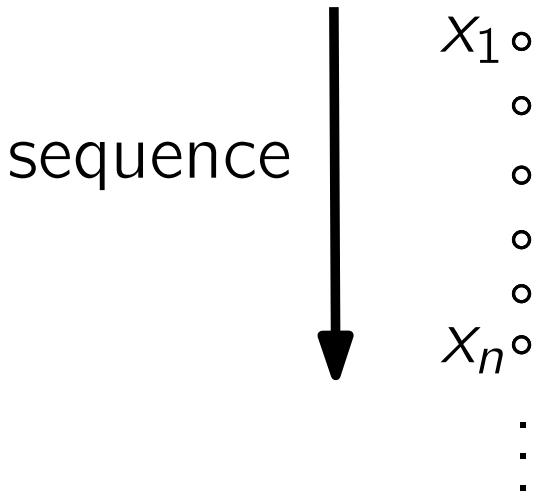
input aligned in 1D (speech, text) 2D (images) or 3D (videos)

Convolutional Nets

input aligned in 1D (speech, text) 2D (images) or 3D (videos) and the labels are invariant on translation

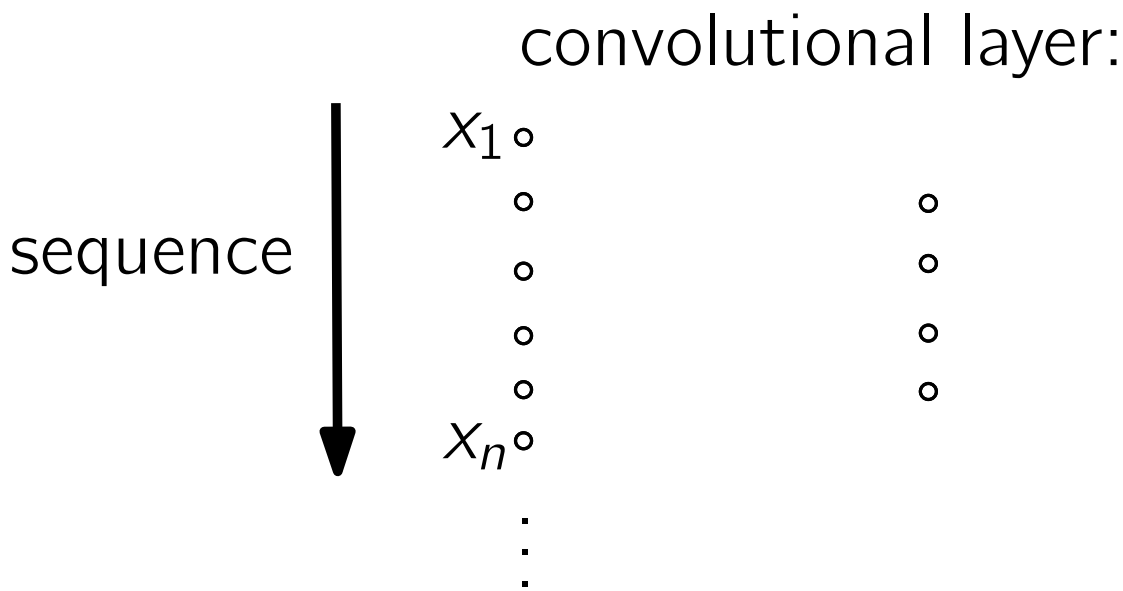
Convolutional Nets

input aligned in 1D (speech, text) 2D (images) or 3D (videos) and the labels are invariant on translation



Convolutional Nets

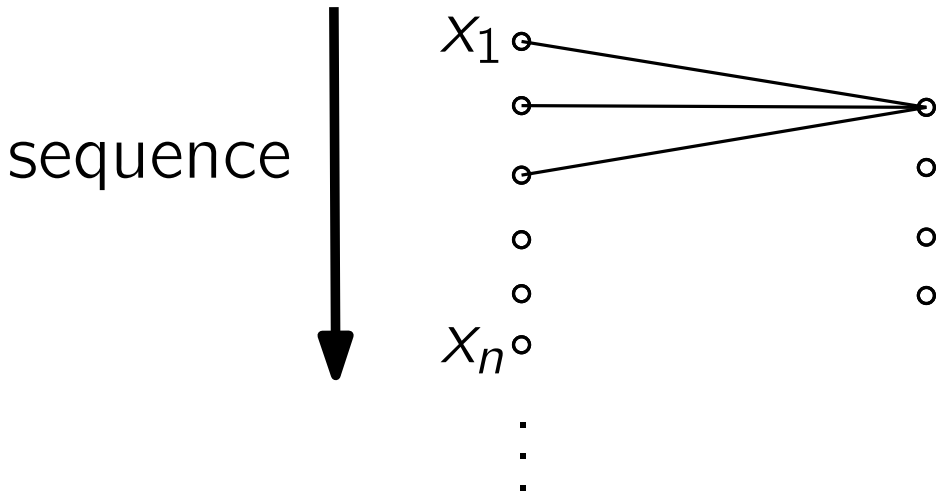
input aligned in 1D (speech, text) 2D (images) or 3D (videos) and the labels are invariant on translation



Convolutional Nets

input aligned in 1D (speech, text) 2D (images) or 3D (videos) and the labels are invariant on translation

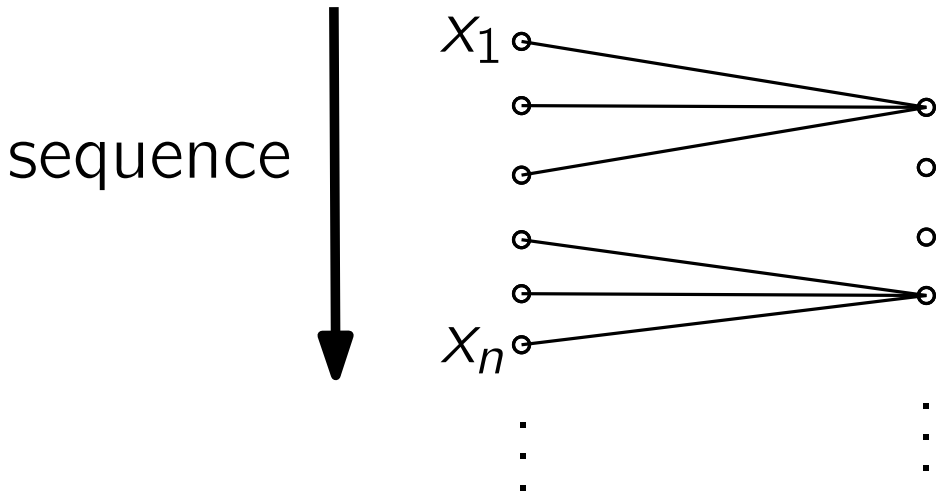
convolutional layer:



Convolutional Nets

input aligned in 1D (speech, text) 2D (images) or 3D (videos) and the labels are invariant on translation

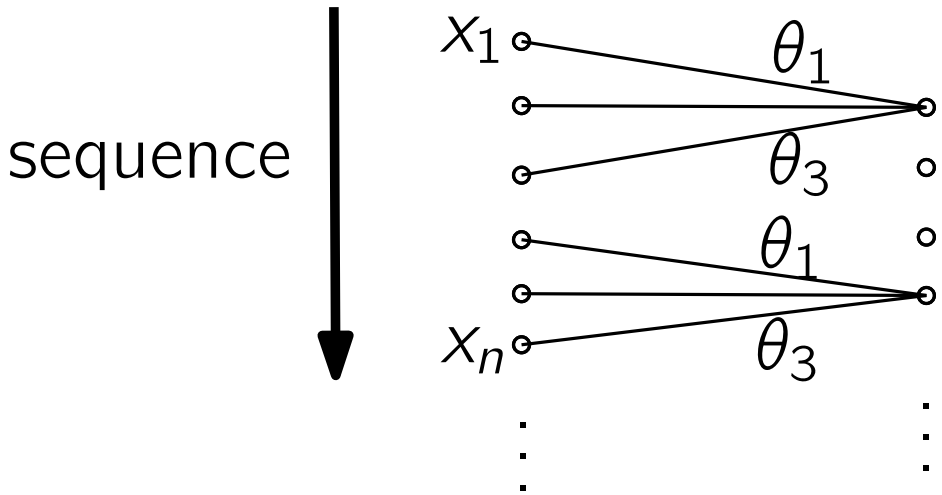
convolutional layer:



Convolutional Nets

input aligned in 1D (speech, text) 2D (images) or 3D (videos) and the labels are invariant on translation

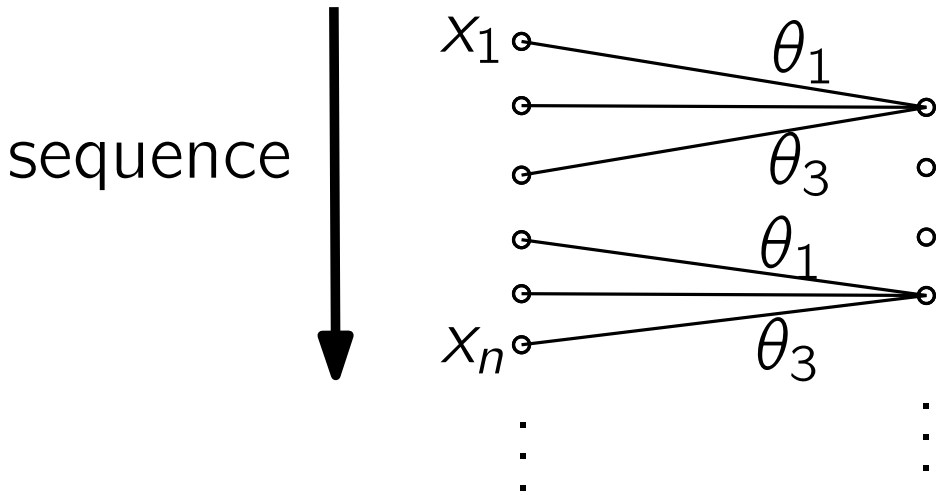
convolutional layer:



Convolutional Nets

input aligned in 1D (speech, text) 2D (images) or 3D (videos) and the labels are invariant on translation

convolutional layer:

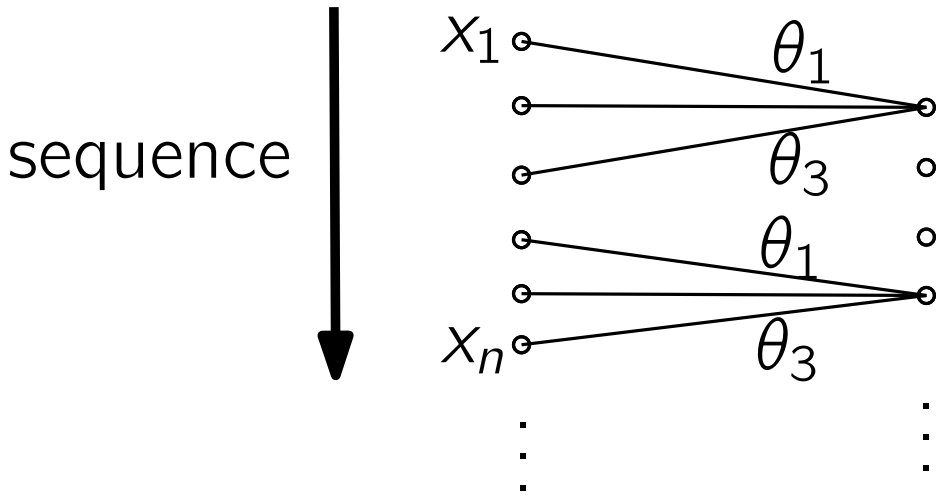


- convolutional layers keep variable-sized representation

Convolutional Nets

input aligned in 1D (speech, text) 2D (images) or 3D (videos) and the labels are invariant on translation

convolutional layer:



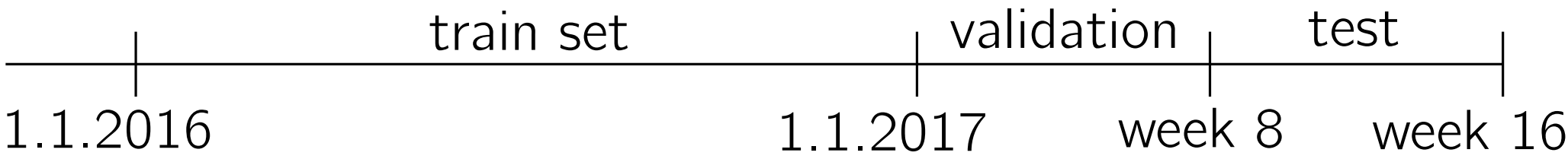
- convolutional layers keep variable-sized representation
- operations like max or average to get a fixed size

Dataset

- PE (Portable Executable) files from $\sim$ 16 months (71 mil)
- Unpacked (32 mil)
- Between 12kB and 1/2 MB (19 mil)

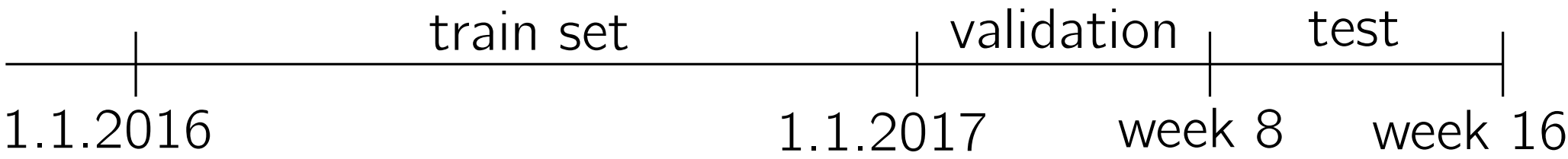
Dataset

- PE (Portable Executable) files from $\sim$ 16 months (71 mil)
- Unpacked (32 mil)
- Between 12kB and 1/2 MB (19 mil)
- Temporal split:



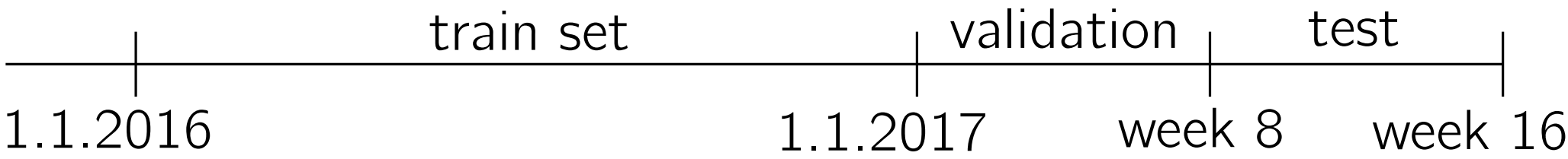
Dataset

- PE (Portable Executable) files from $\sim$ 16 months (71 mil)
- Unpacked (32 mil)
- Between 12kB and 1/2 MB (19 mil)
- Temporal split:



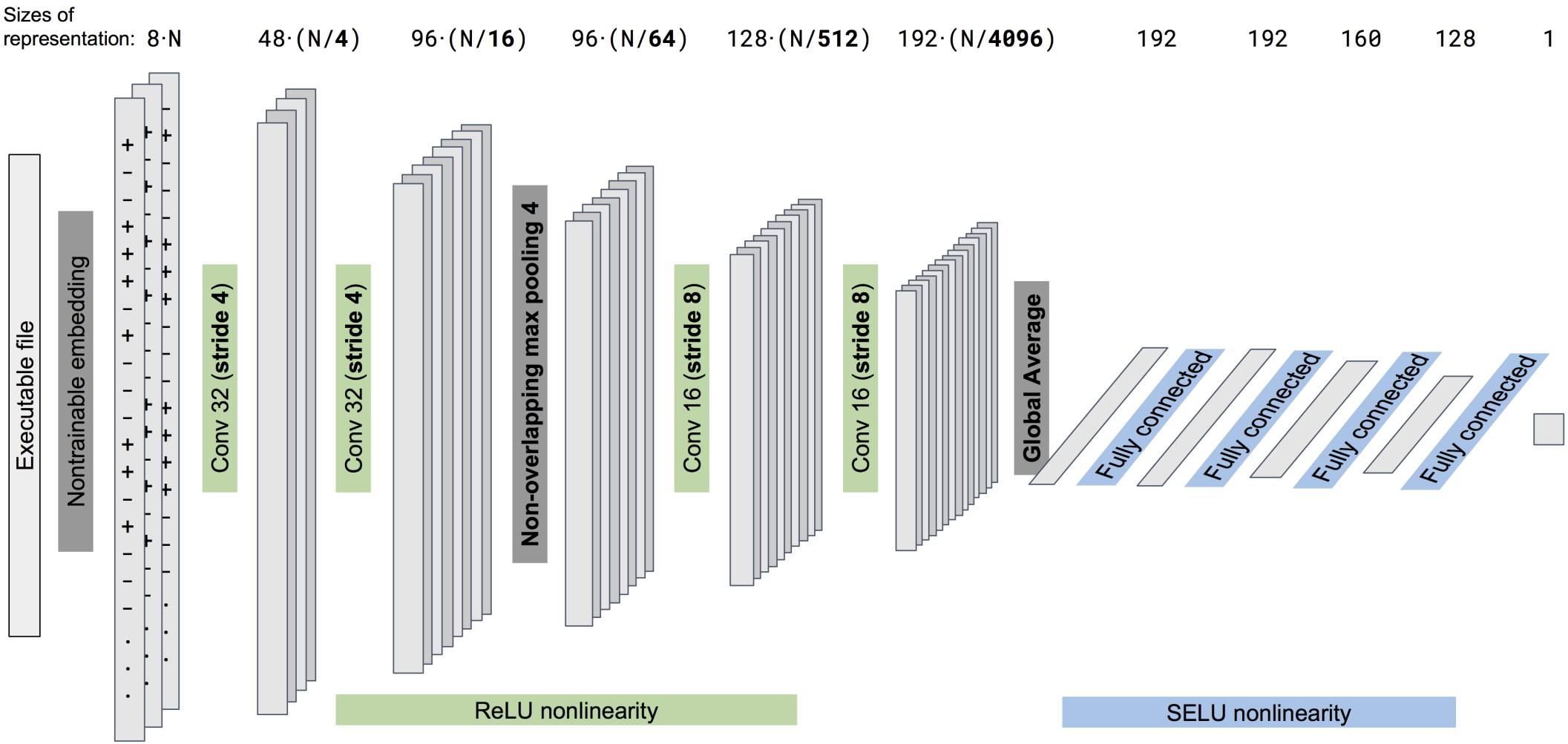
Dataset

- PE (Portable Executable) files from $\sim$ 16 months (71 mil)
- Unpacked (32 mil)
- Between 12kB and 1/2 MB (19 mil)
- Temporal split:



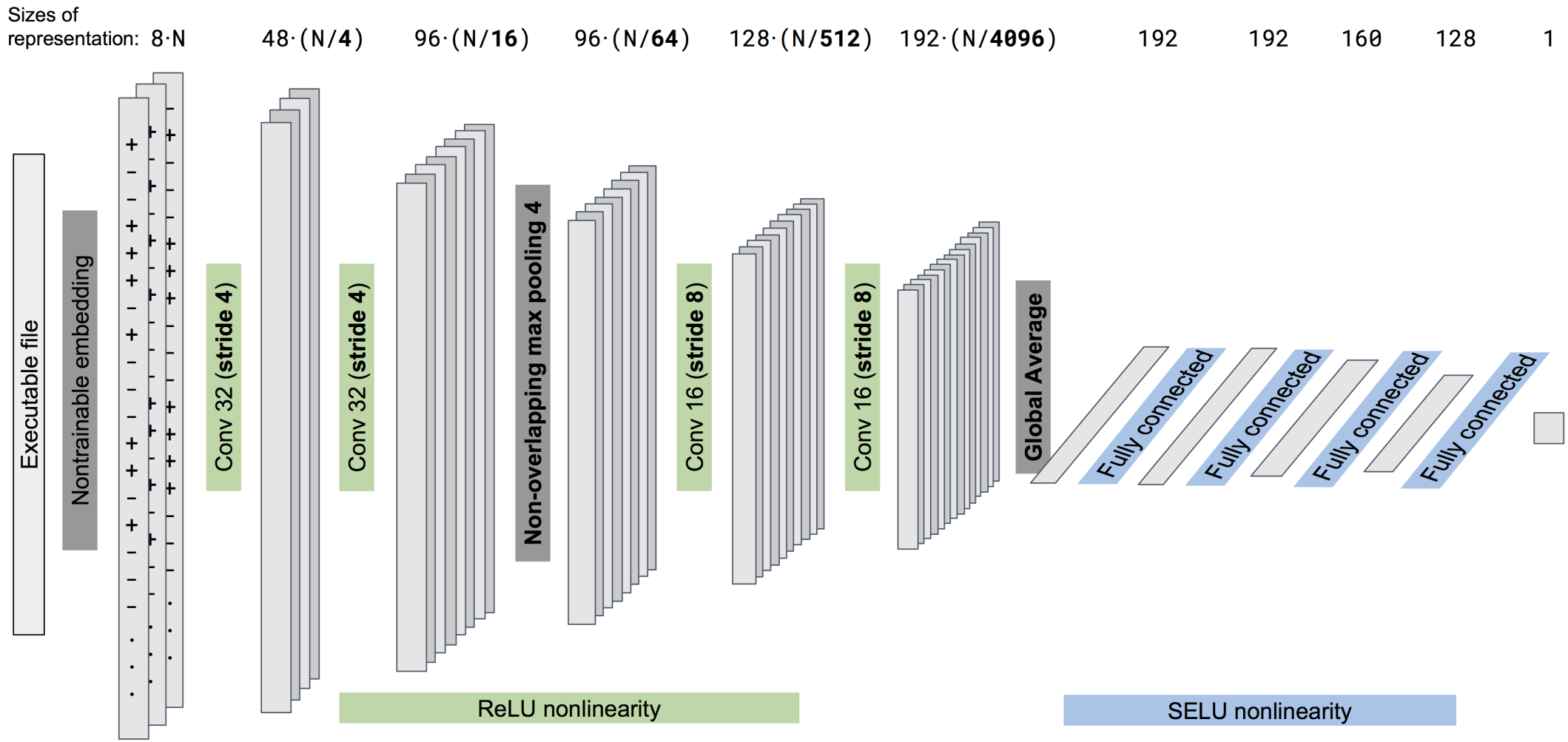
- 33% malware when repeating every clean executable twice (we need to punish false positives more)

Architecture



827969 trainable parameters with weight decay $3 \cdot 10^{-7}$

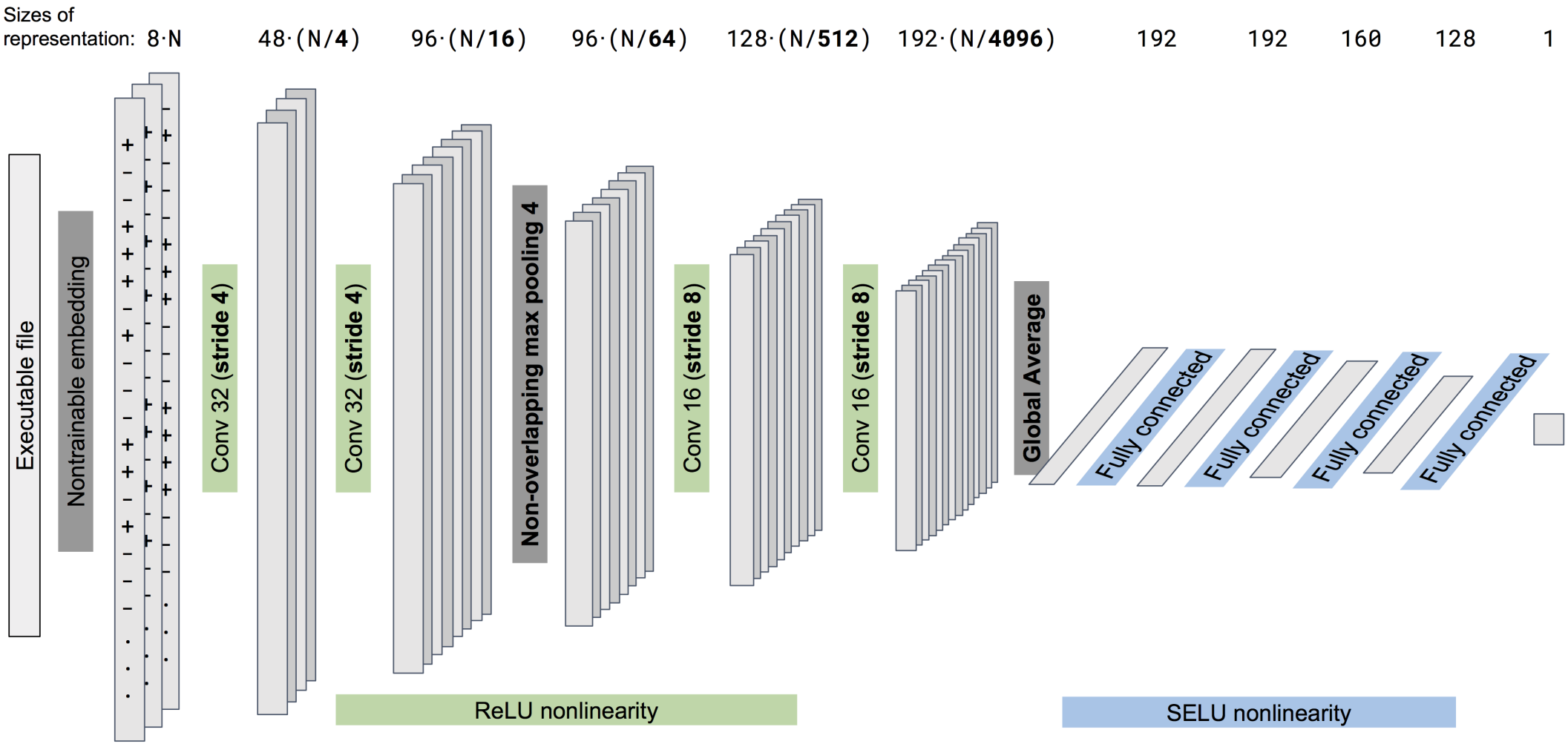
Architecture



827969 trainable parameters with weight decay $3 \cdot 10^{-7}$

- strides improve performance and speed (powers of two)

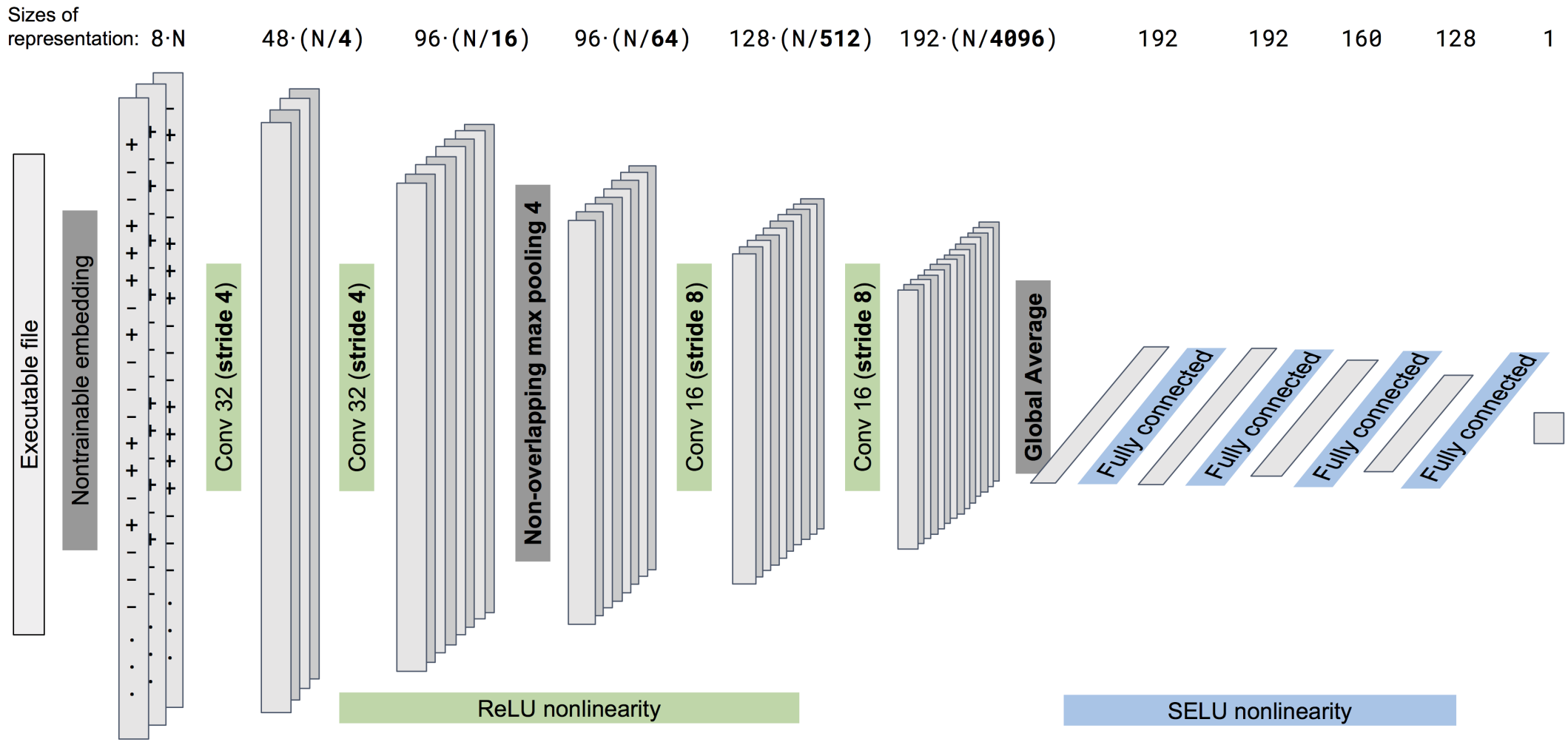
Architecture



827969 trainable parameters with weight decay $3 \cdot 10^{-7}$

- strides improve performance and speed (powers of two)
- only ℓ_2 regularization used to stabilize training

Architecture



827969 trainable parameters with weight decay $3 \cdot 10^{-7}$

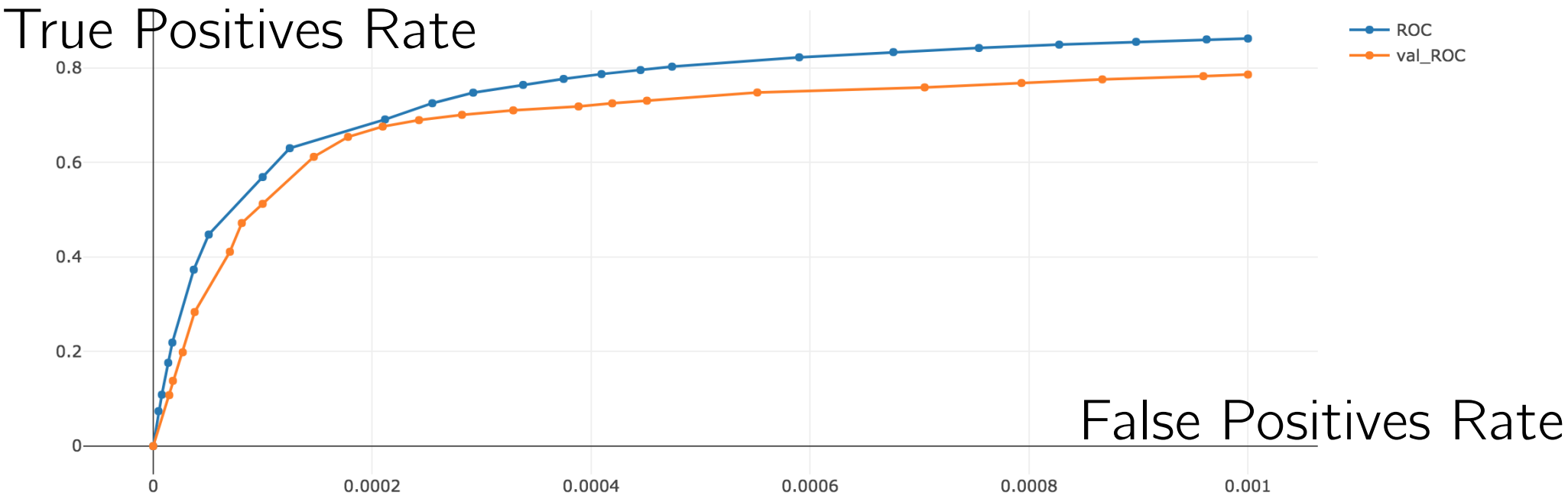
- strides improve performance and speed (powers of two)
- only ℓ_2 regularization used to stabilize training
- standard init, Adam, cross-entropy loss

Evaluation

- Score: area under the receiver operator curve restricted to $[0, 0.001]$ (AUC $\mathrm{ROC}_{[0,0.001]}$)

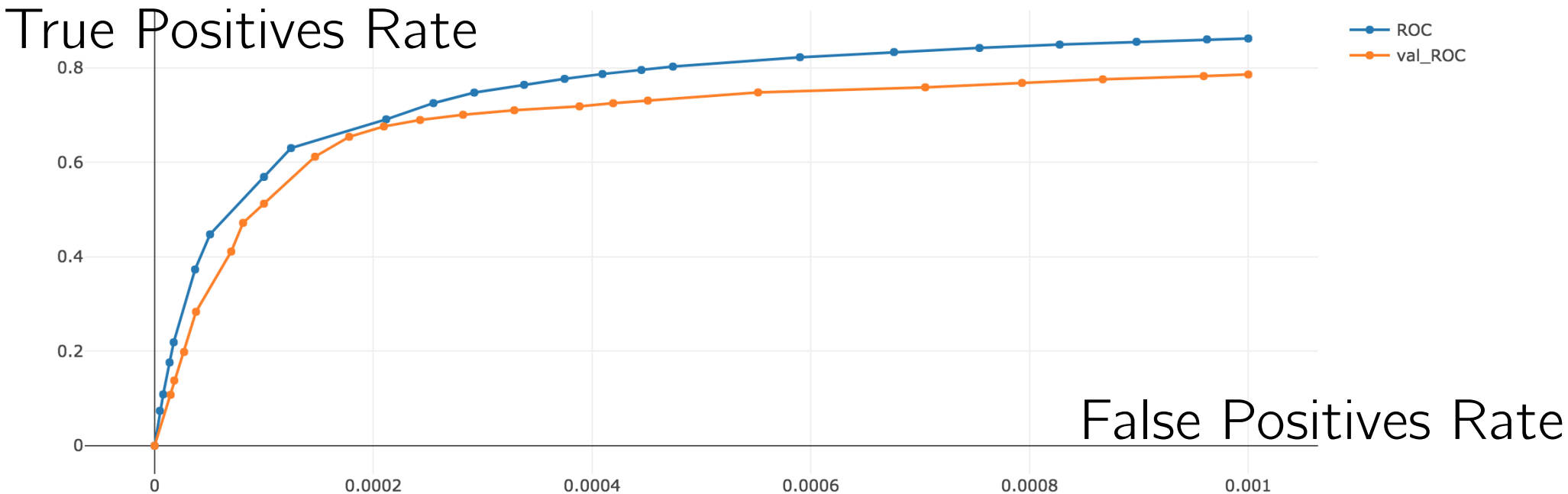
Evaluation

- Score: area under the receiver operator curve restricted to $[0, 0.001]$ (AUC $\text{ROC}_{[0,0.001]}$)

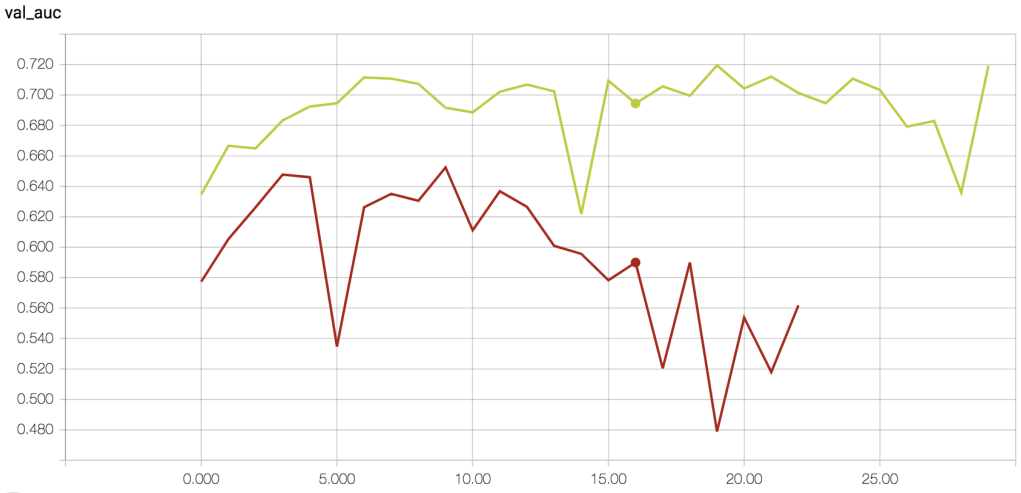


Evaluation

- Score: area under the receiver operator curve restricted to $[0, 0.001]$ (AUC $\text{ROC}_{[0,0.001]}$)



- Training time: 2-3 days



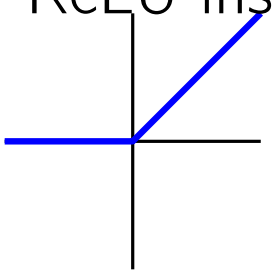
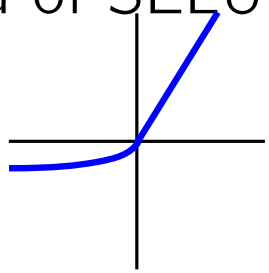
Results (choose the right score!)

	AUC $\text{ROC}_{[0,0.001]}$
Our architecture	0.667 ± 0.019

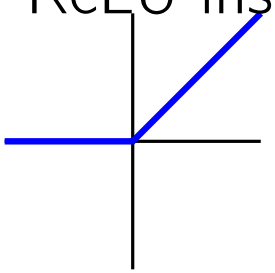
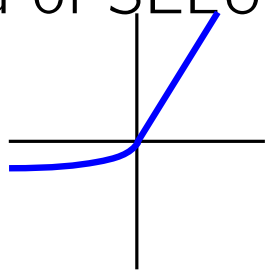
Results (choose the right score!)

	AUC $\text{ROC}_{[0,0.001]}$
Our architecture	0.667 ± 0.019
Max instead of Avg	-20% but better accuracy and x-entropy

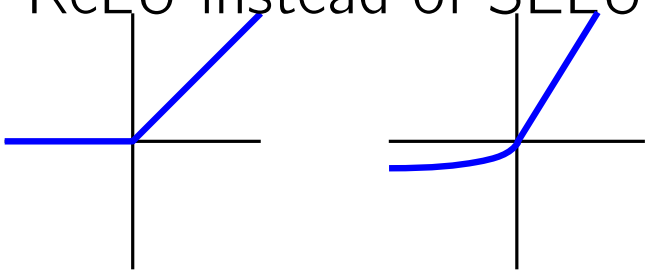
Results (choose the right score!)

	AUC $\text{ROC}_{[0,0.001]}$
Our architecture	0.667 ± 0.019
Max instead of Avg	-20% but better accuracy and x-entropy
ReLU instead of SELU  	-4% but better x-entropy

Results (choose the right score!)

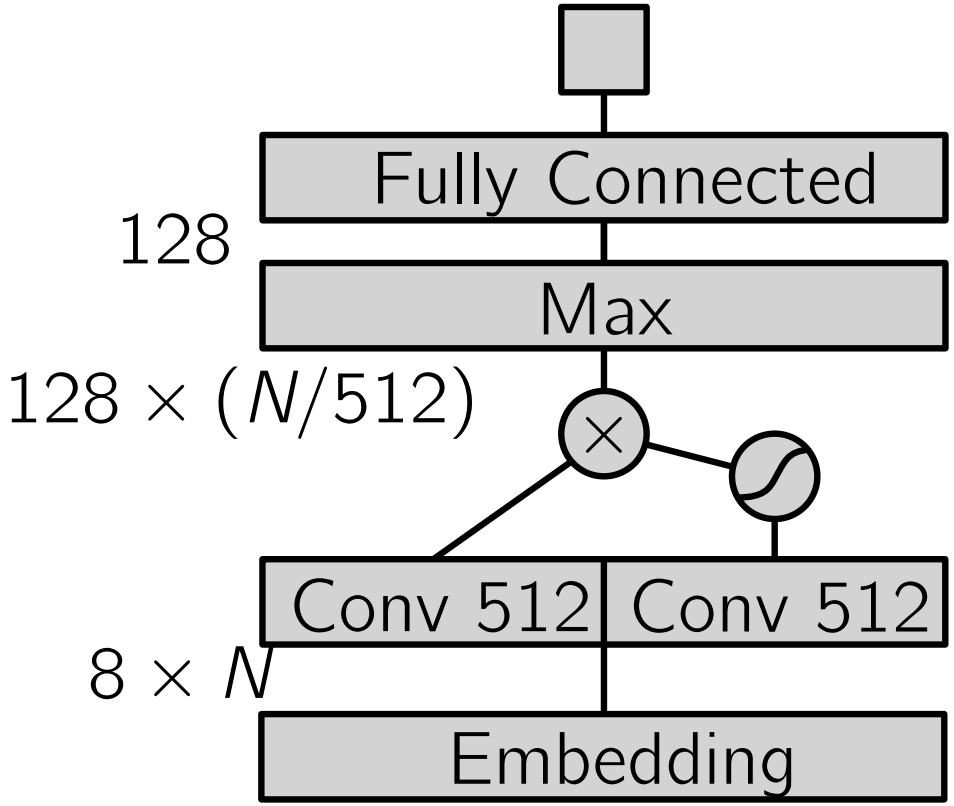
	AUC $\text{ROC}_{[0,0.001]}$
Our architecture	0.667 ± 0.019
Max instead of Avg	-20% but better accuracy and x-entropy
ReLU instead of SELU  	-4% but better x-entropy

Results (choose the right score!)

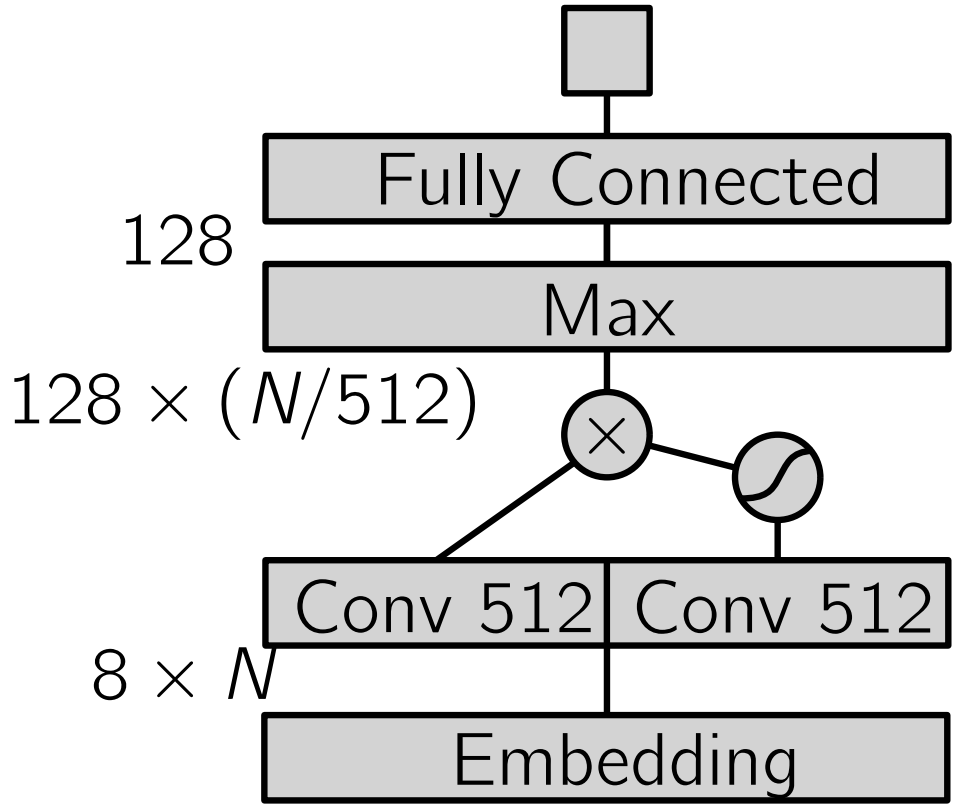
	AUC $\mathrm{ROC}_{[0,0.001]}$
Our architecture	0.667 ± 0.019
Max instead of Avg	-20% but better accuracy and x-entropy
ReLU instead of SELU 	-4% but better x-entropy
Strides 3,5,7,9 instead of 4,4,8,8	-8%

Competing architecture (amount of data matters!)

Competing architecture (amount of data matters!)



Competing architecture (amount of data matters!)



	AUC $\text{ROC}_{[0,0.001]}$
Our architecture	0.667 ± 0.019
MalConv (competitor)	~ 0.62

Automatic vs. hand-crafted features

Avast's latest ML system uses 538 in-house hand-crafted features

Automatic vs. hand-crafted features

Avast's latest ML system uses 538 in-house hand-crafted features

feed to a 5-layer feedforward net (same dataset):

Automatic vs. hand-crafted features

Avast's latest ML system uses 538 in-house hand-crafted features

feed to a 5-layer feedforward net (same dataset):

	AUC $\mathrm{ROC}_{[0,0.001]}$
convolution features	~ 0.667
hand-crafted features	~ 0.71 but worse accuracy and x-entropy

Automatic vs. hand-crafted features

Avast's latest ML system uses 538 in-house hand-crafted features

feed to a 5-layer feedforward net (same dataset):

	AUC $\text{ROC}_{[0,0.001]}$
convolution features	~ 0.667
hand-crafted features	~ 0.71 but worse accuracy and x-entropy
ensamble	~ 0.75 and better accuracy and x-entropy

Automatic vs. hand-crafted features

Avast's latest ML system uses 538 in-house hand-crafted features

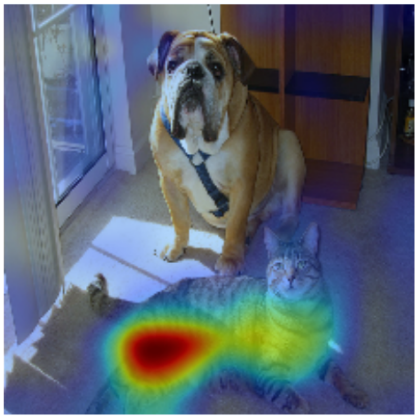
feed to a 5-layer feedforward net (same dataset):

	AUC $\text{ROC}_{[0,0.001]}$
convolution features	~ 0.667
hand-crafted features	~ 0.71 but worse accuracy and x-entropy
ensamble	~ 0.75 and better accuracy and x-entropy

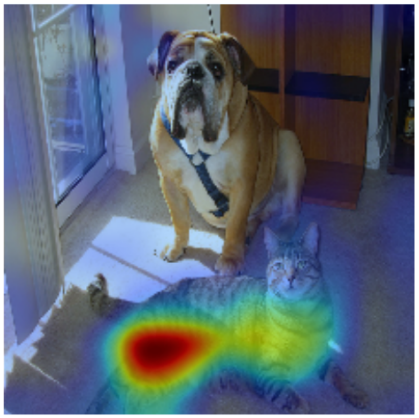
Dataset made it easier for conv net to compete, but..

Explainability

- grad-CAM (Class Activation Map):



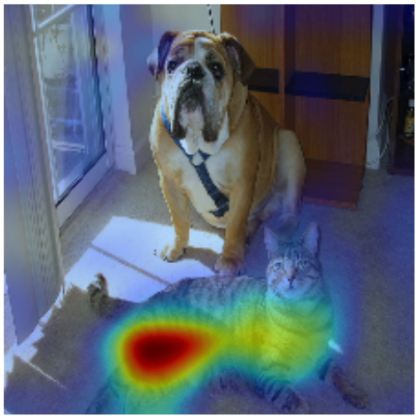
Explainability



- grad-CAM (Class Activation Map):

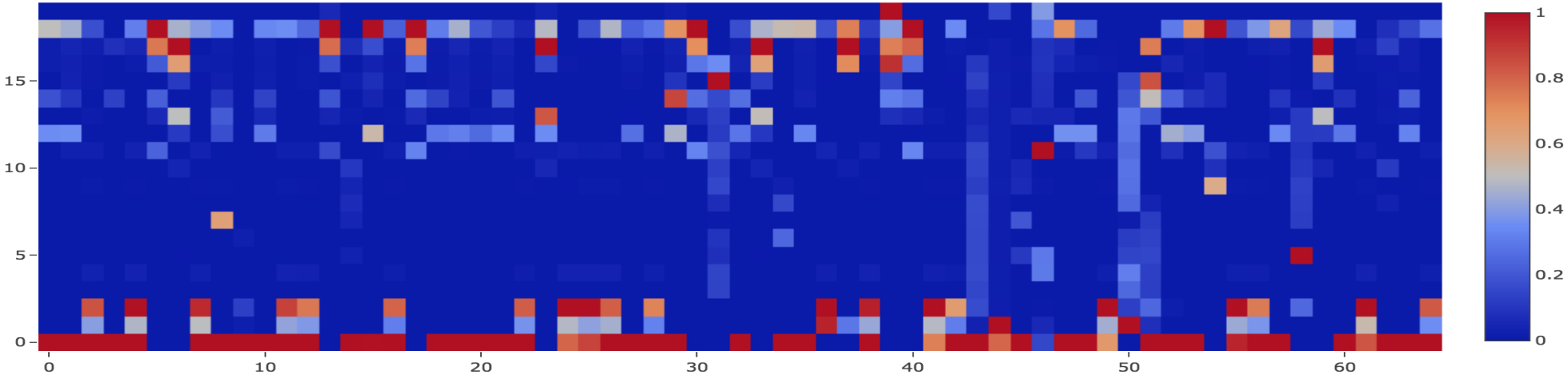
  applicable to any conv net

Explainability

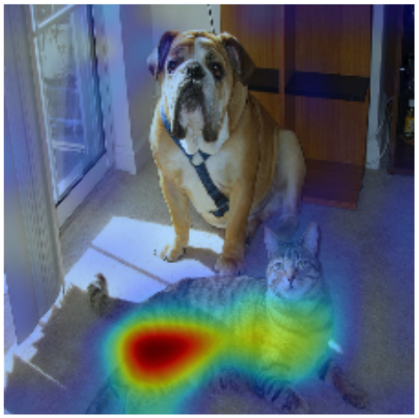


- grad-CAM (Class Activation Map):

  applicable to any conv net

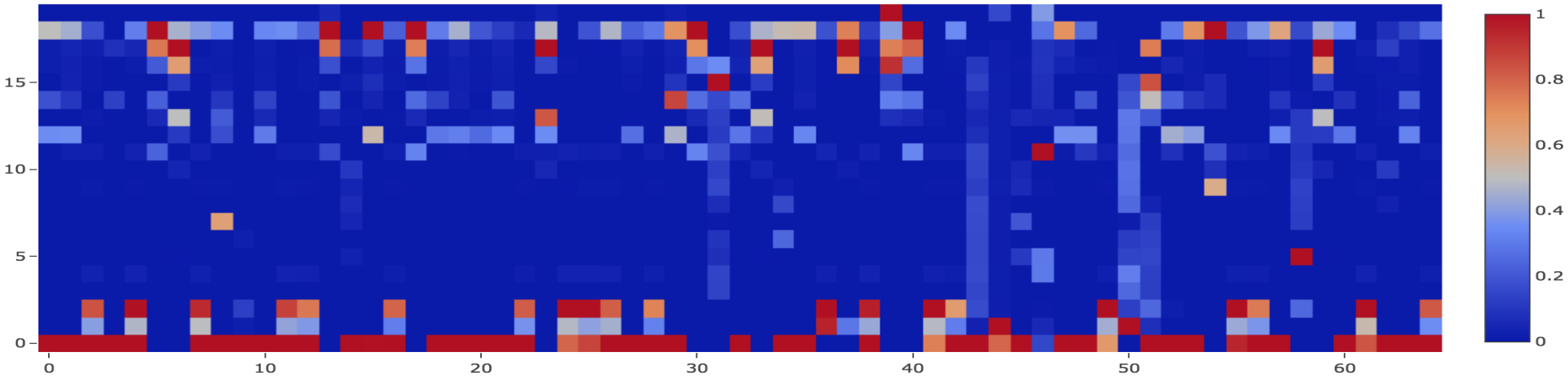


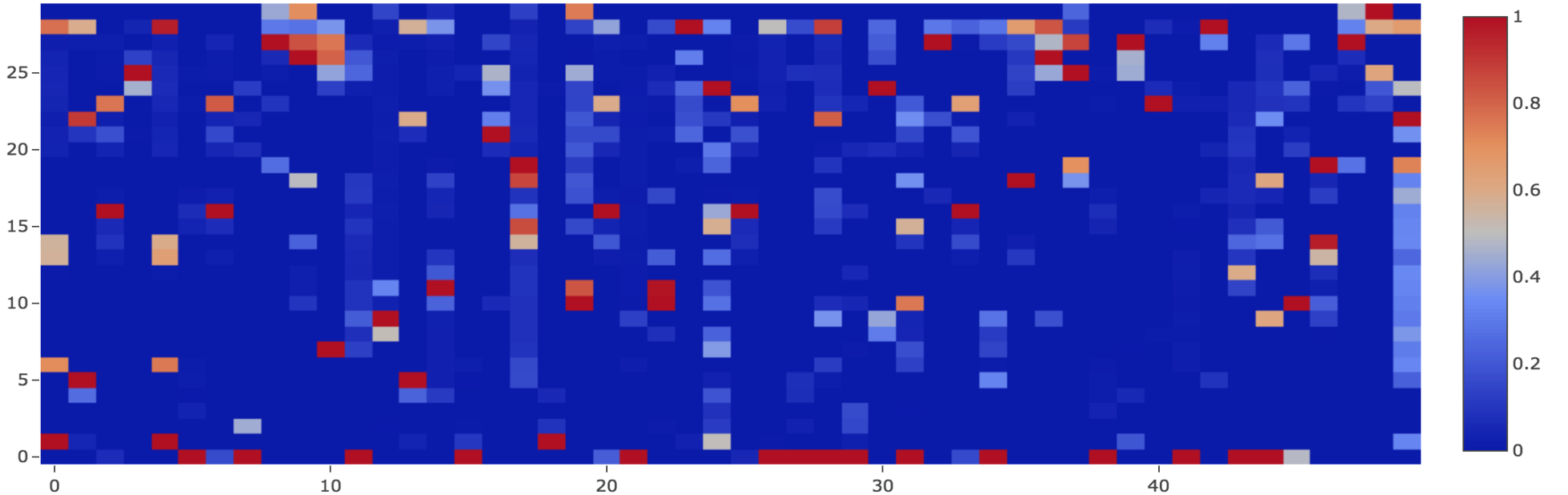
Explainability



- grad-CAM (Class Activation Map):

  applicable to any conv net





Problems/Future work

- Talk to analyst.

Problems/Future work

- Talk to analyst.
- Reduce the training time.

Problems/Future work

- Talk to analyst.
- Reduce the training time.
- Advanced architectures (separable or gated convolutions)
- Other input: Android files, emulator log

Problems/Future work

- Talk to analyst.
- Reduce the training time.
- Advanced architectures (separable or gated convolutions)
- Other input: Android files, emulator log

Problems/Future work

- Talk to analyst.
- Reduce the training time.
- Advanced architectures (separable or gated convolutions)
- Other input: Android files, emulator log
- Tune the "restricted AUC" more. Adapt x-entropy loss?

- Large-scale application of conv nets to a new domain (**data volume matters**)

- Large-scale application of conv nets to a new domain (**data volume matters**)
- Training for low false positives (**correct score matters**)

- Large-scale application of conv nets to a new domain (**data volume matters**)
- Training for low false positives (**correct score matters**)