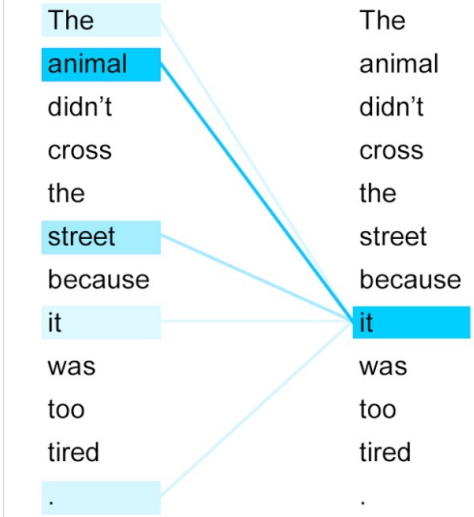
Separable Self-Attention for Generative Models

Utkarsh Singhal

Self-Attention is everywhere

- PixelSNAIL
- Transformers/OpenGPT-2
- Image/sparse transformers
- Self-attention GANs, BigGANs





Revisiting self-attention

- Self-attention is defined by the following equation:

$$y_i = \sum_j f(q_i, k_j) v_j$$

- F is the similarity between query location i and key at location j. Common choice is “Dot-product attention”:

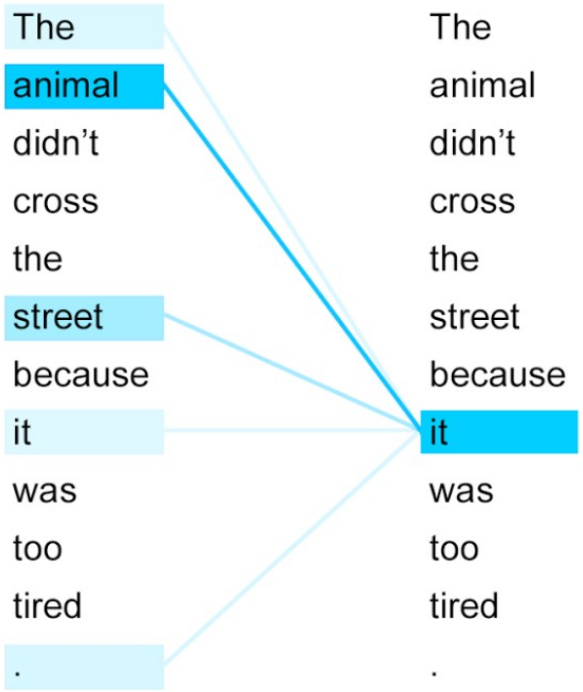
$$f(q_i, k_j) = e^{q_i^T k_j}$$

Problems with Self-Attention: Complexity

- Computing $f(q_i, k_j)$ requires comparing every i with every j. O(n^2) complexity!
- 3.2x larger image dimensions means 100x more memory required
- Forces low-res images and short sentences (causing “context fragmentation”)

Problems with Self-Attention: Linearity

- Every word is processed in isolation; no notion of context or conditional dependence
- How do you count?
- How do you handle double negatives? (e.g. “I don’t dislike cats”)



Enter kernel approximations

Main idea: Kernels are inner products in some embedding space. We can try to approximate them with dot products of another embedding.

- Random Fourier Features
- Random ReLU Features
- Tensor Sketching
- MacLaurin Features
- Random Binning Features
- and much more!

$$k(x, y) \approx \phi(x)^T \phi(y)$$

What if we could separate f?

- Let $f(q_i, k_j) = e^{q_i^T k_j} \approx \phi(q_i)^T \phi(k_j)$
- Then we can re-arrange the equation:

$$y_i = \sum_j \phi(q_i)^T \phi(k_j) v_j = \left(\sum_j v_j \phi(k_j)^T \right) \phi(q_i)$$

- Can be interpreted as computing the cross-covariance matrix of v & k, and using it as a weight for 1x1 convolutions on top of q.
- Can also be interpreted as a hebbian associative memory like “Fast Weights” (from Ba and Hinton, 2016)

Computational Advantages

- O(n) memory consumption.

$$y_i = \sum_j \phi(q_i)^T \phi(k_j) v_j = \left(\sum_j v_j \phi(k_j)^T \right) \phi(q_i)$$

- On my GPU, can easily process multi-megapixel images while traditional self-attention struggles at 100x100 images

Representational Advantages

- Can express higher-order attention

$$y_i = \sum_a \sum_b f(k_a, k_b, q_i)(v_a \odot v_b) = \sum_a \sum_b (k_a \odot k_b \odot q_i)^T \mathbf{1}(v_a \odot v_b)$$

- Equivalent to using a similarity tensor.
- Allows location a to influence the attention weight for location b through multiplicative/gating operations.
- Allows self-attention to count!
- Similar to Smolensky/Schmidhuber's tensor-product representations for reasoning

Toy Experiment Results: Learning XOR

- **Task:** 2 “pixel” input, 2 “pixel” output, 64-dim key/query vectors. Output should be [1,1] or [-1,-1] depending on XOR of inputs.
- Softmax attention results in MSE of 0.66
- First-order separable attention results in MSE of 0.68
- Second-order separable attention results in MSE of 1e-3

Toy Experiment Results: Learning to count

- **Task:** 3 “pixel” input, 3 “pixel” output. Output is [1,1,1] iff >=2 pixels in the input are 1, otherwise the output is [-1,-1,-1]
- Softmax attention results in MSE of 0.23
- First-order separable attention results in MSE of 0.23
- Second-order separable attention results in MSE of 1e-3

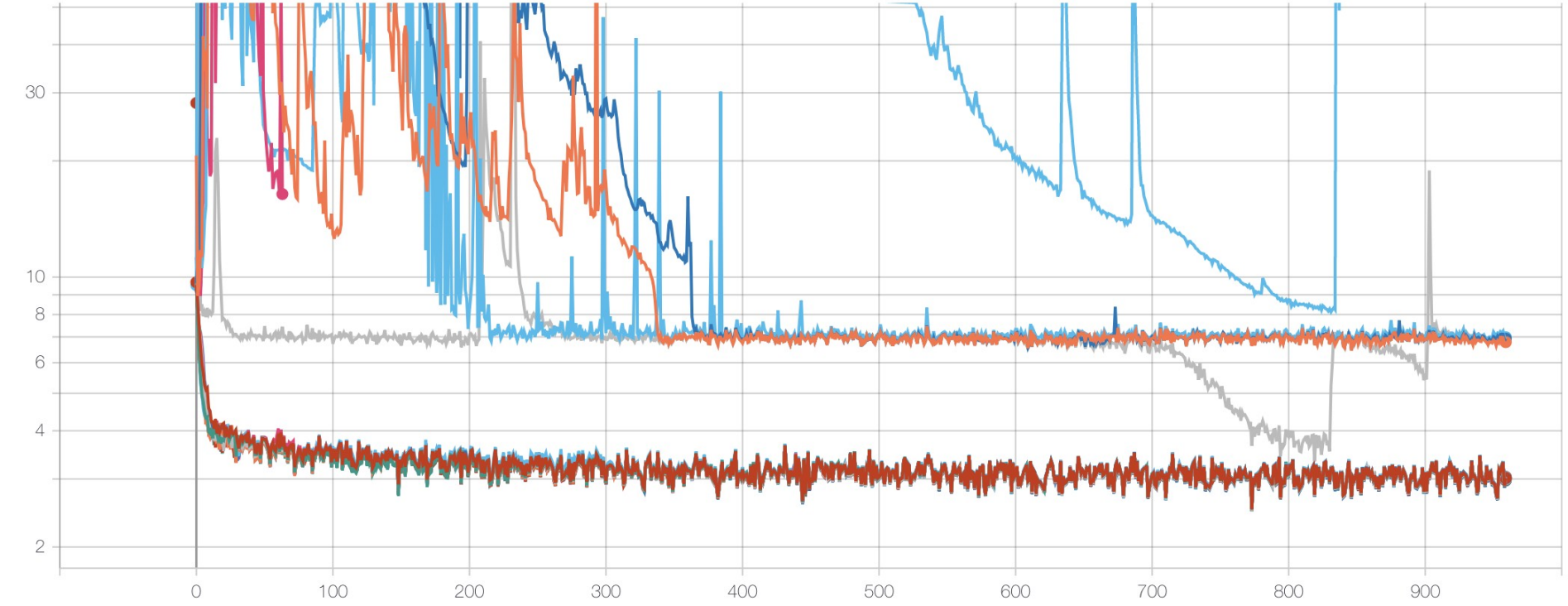
Experiments on PixelSNAIL

- Can recover the paper’s original result on both CIFAR and ImageNet by replacing e^(k.v) with abs(k).abs(v)
- What other types of non-linearities work?

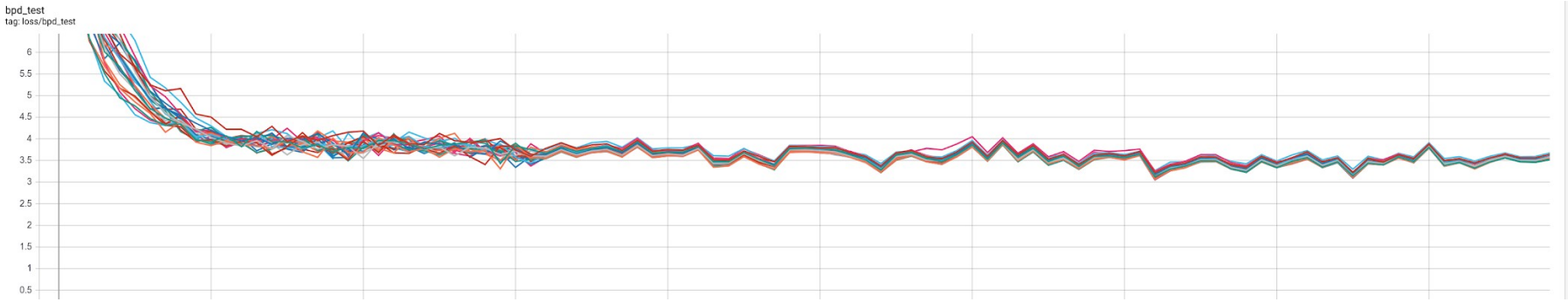
Experiments on PixelSNAIL

- 36 experiments in total:
- Pre-attention normalization: None, Squash, L2
- Random Projections? No, Yes (128)
- Activation function: None, abs, relu, tanh, sigmoid, rffm

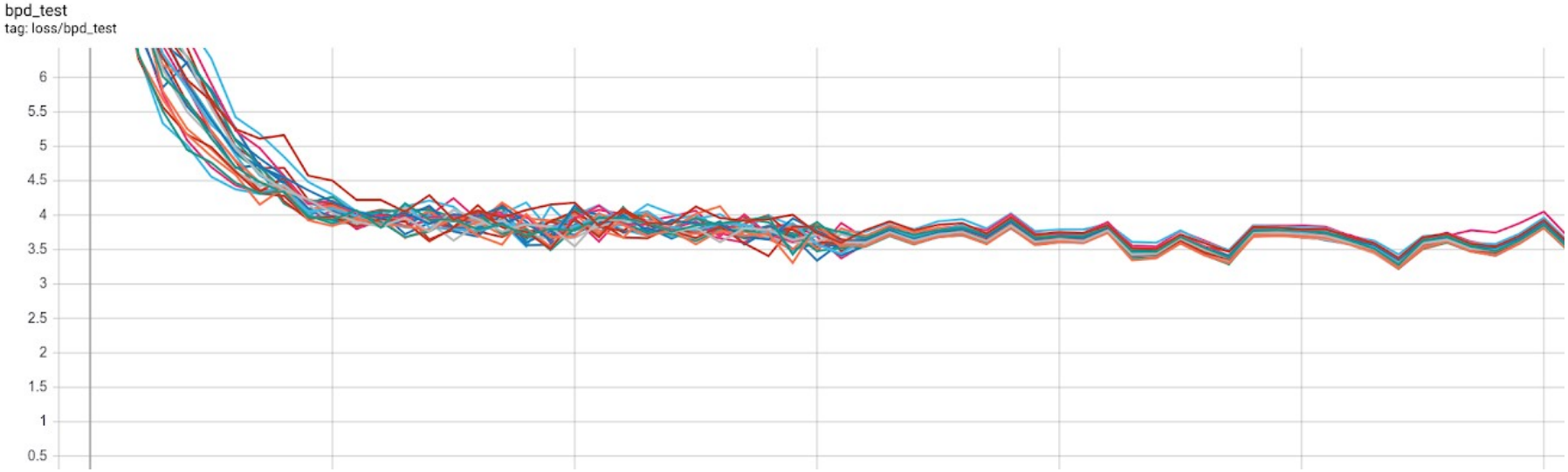
Experiments on PixelSNAIL CIFAR: Results



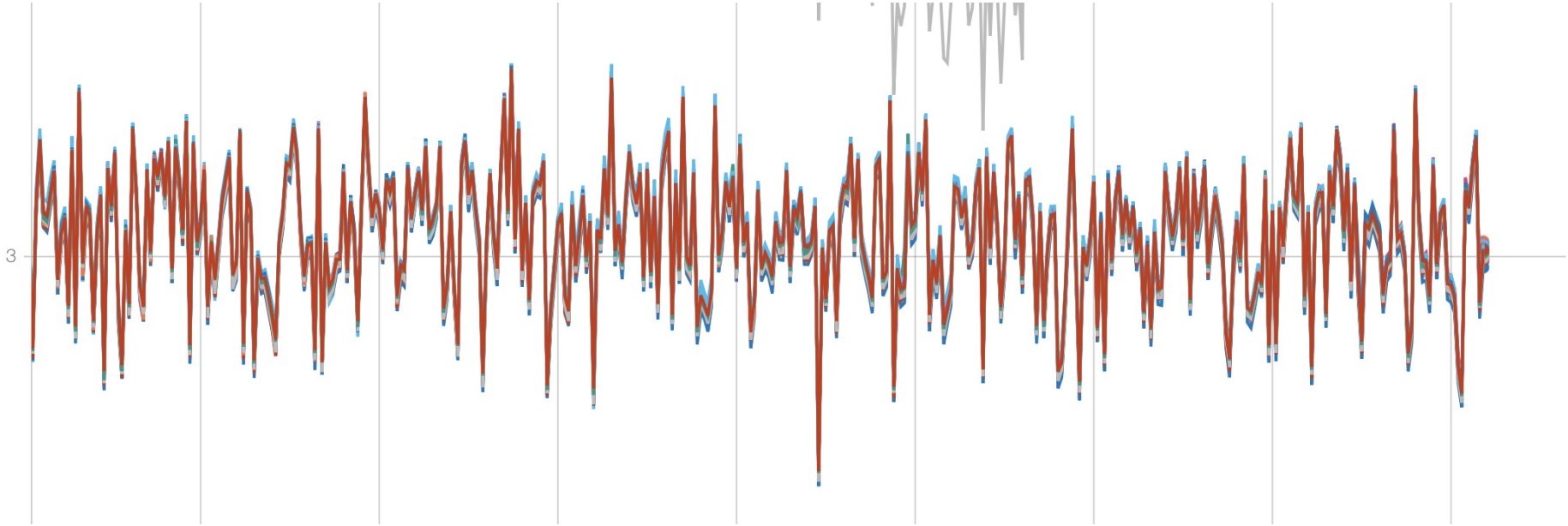
Experiments on PixelSNAIL: Results



Experiments on PixelSNAIL: Results



Experiments on PixelSNAIL: Results



Experiments on PixelSNAIL: Conclusions

- Why so much correlation? I don’t know
- 24 out of 36 configurations made it close to 3 bits/dim
- Without a pre-activation normalization, only relu, abs, and sigmoid (the non-negative activations) get to 3 bpd
- Tanh seems to require random projections and a normalization
- RFFM seems to require squashing activation

Experiments on PixelSNAIL: Disclaimers

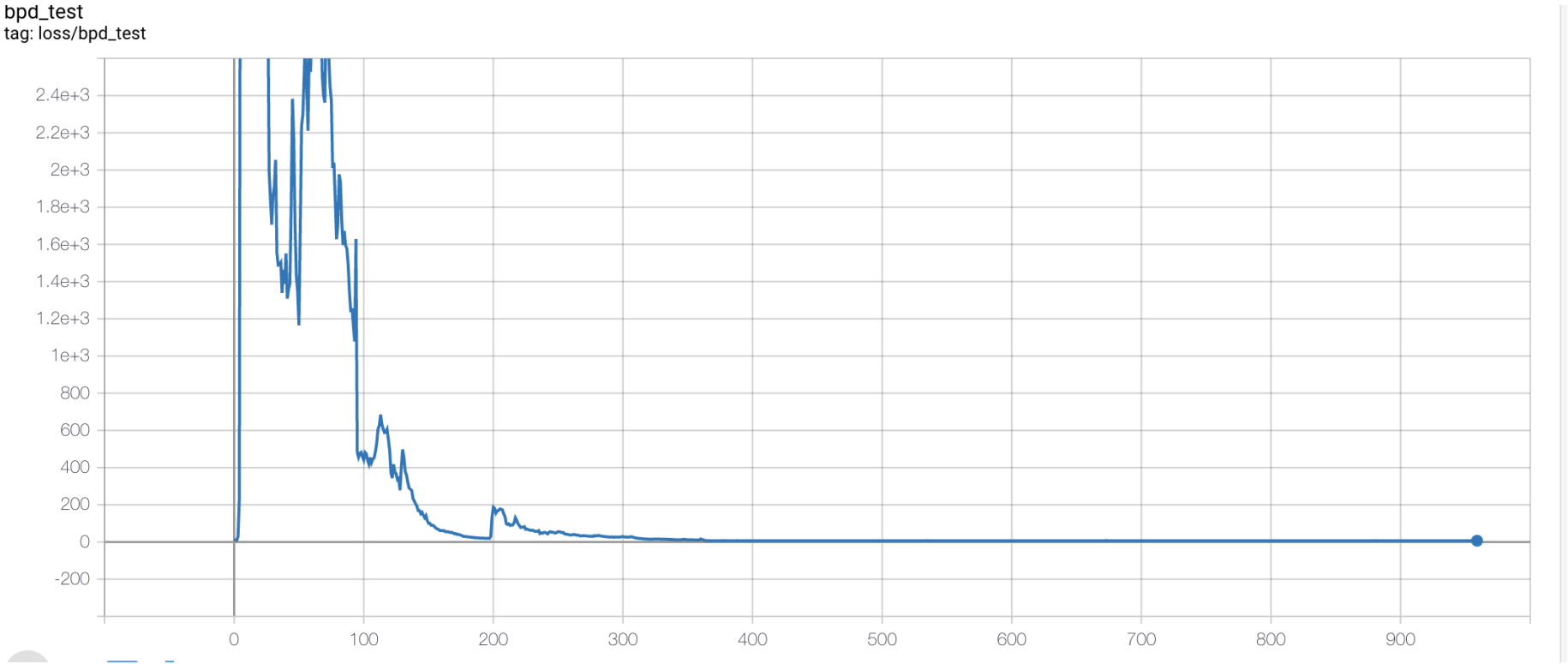
- No hyperparameter tuning at all
- Didn’t train for nearly long enough: state of the art result took many days

Experiments on a text parser

- Tried Nikita Kitaev’s “Berkeley Neural Parser”
- This form of attention currently leads to instability

Thanks!

No normalization/activation/random-projection



Configurations that got to 3 bpd

norm: l2	act: False	nrproj: 128
norm: squash	act: False	nrproj: 128
norm: l2	act: abs	nrproj: False
norm: squash	act: abs	nrproj: False
norm: squash	act: abs	nrproj: 128
norm: False	act: relu	nrproj: False
norm: False	act: relu	nrproj: 128
norm: l2	act: relu	nrproj: False
norm: l2	act: relu	nrproj: 128
norm: squash	act: sigmoid	nrproj: False
norm: False	act: sigmoid	nrproj: False
norm: squash	act: sigmoid	nrproj: 128
norm: squash	act: rffm	nrproj: 128
norm: l2	act: tanh	nrproj: 128
norm: squash	act: rffm	nrproj: 128
norm: l2	act: tanh	nrproj: 128
norm: squash	act: rffm	nrproj: False
norm: squash	act: tanh	nrproj: 128
norm: squash	act: sigmoid	nrproj: False
norm: squash	act: sigmoid	nrproj: 128
norm: False	act: abs	nrproj: False
norm: False	act: abs	nrproj: 128
norm: l2	act: relu	nrproj: 128
norm: l2	act: relu	nrproj: False