https://unsplash.com/photos/_Sc-HftCbYE

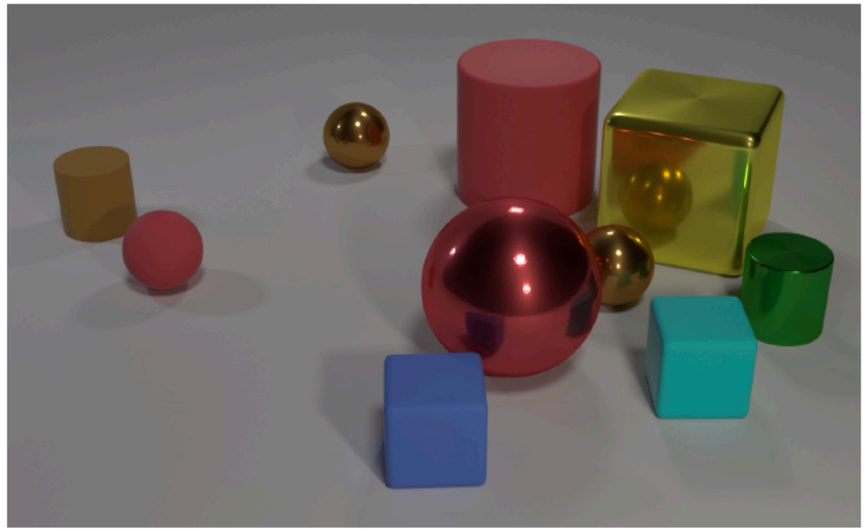
Multiplicative interactions

Alex Fedorov

January 22, 2021

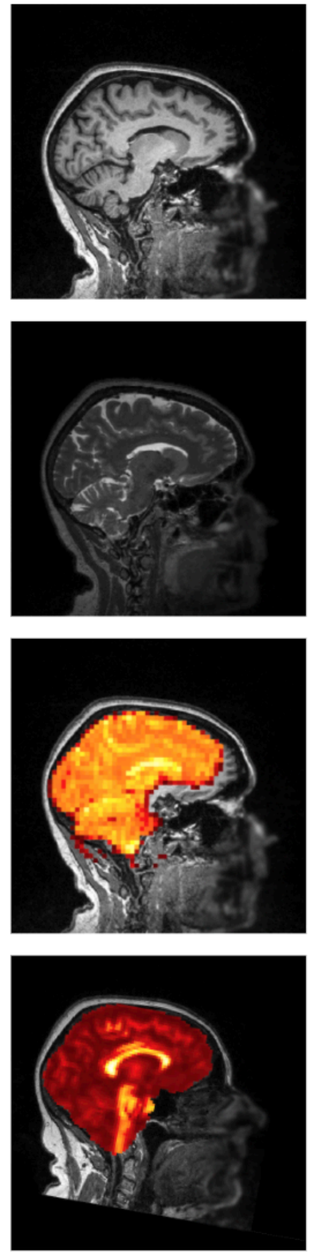
Motivation

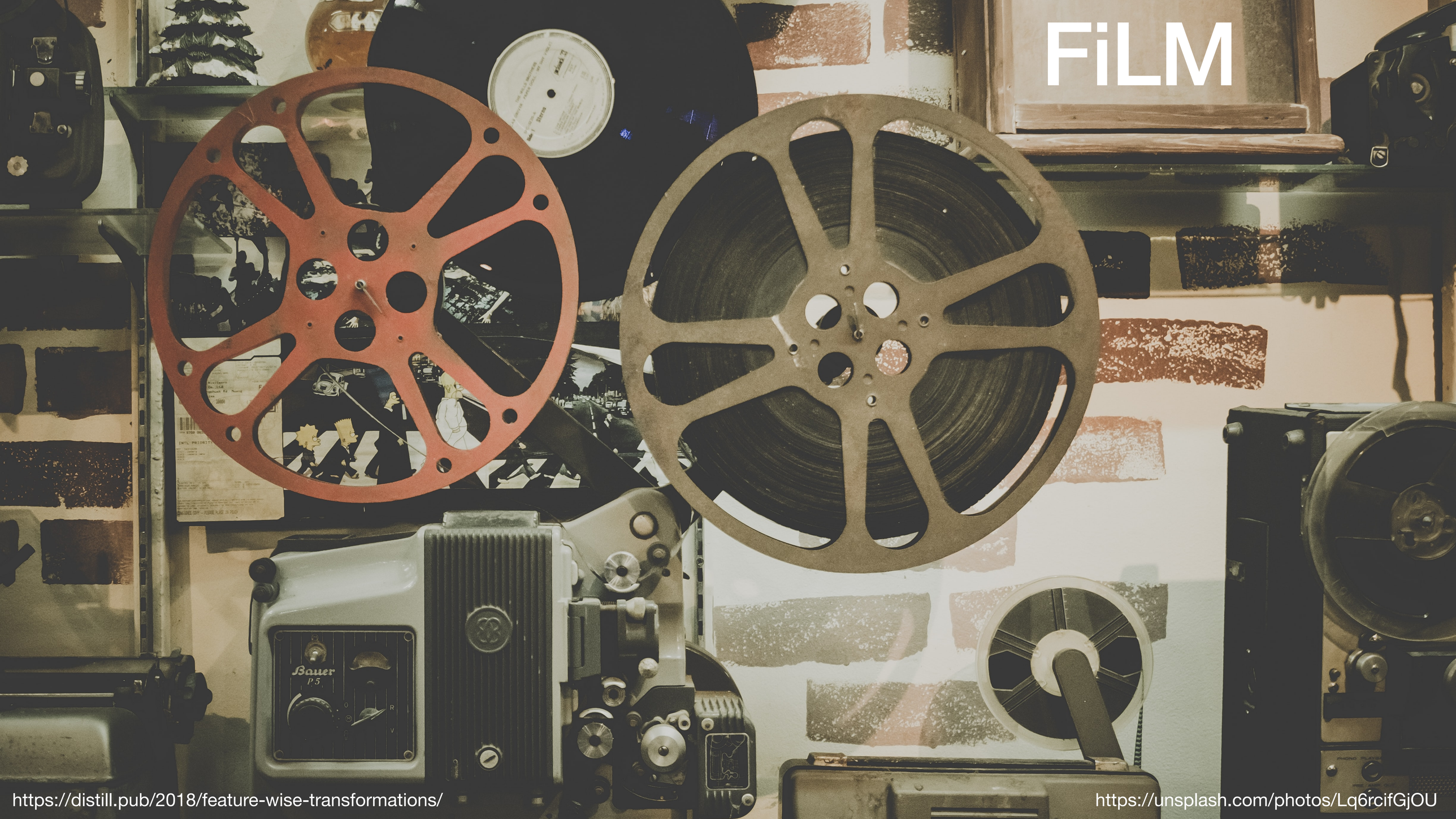
https://weheartit.com/entry/280850620



Q: Are there an equal number of large things and metal spheres? **Q:** What size is the cylinder that is left of the brown metal thing that is left of the big sphere? **Q:** There is a sphere with the same size as the metal cube; is it made of the same material as the small red sphere? **Q:** How many objects are either small cylinders or metal things?

https://towardsdatascience.com/deep-learning-and-visual-question-answering-c8c8093941bc

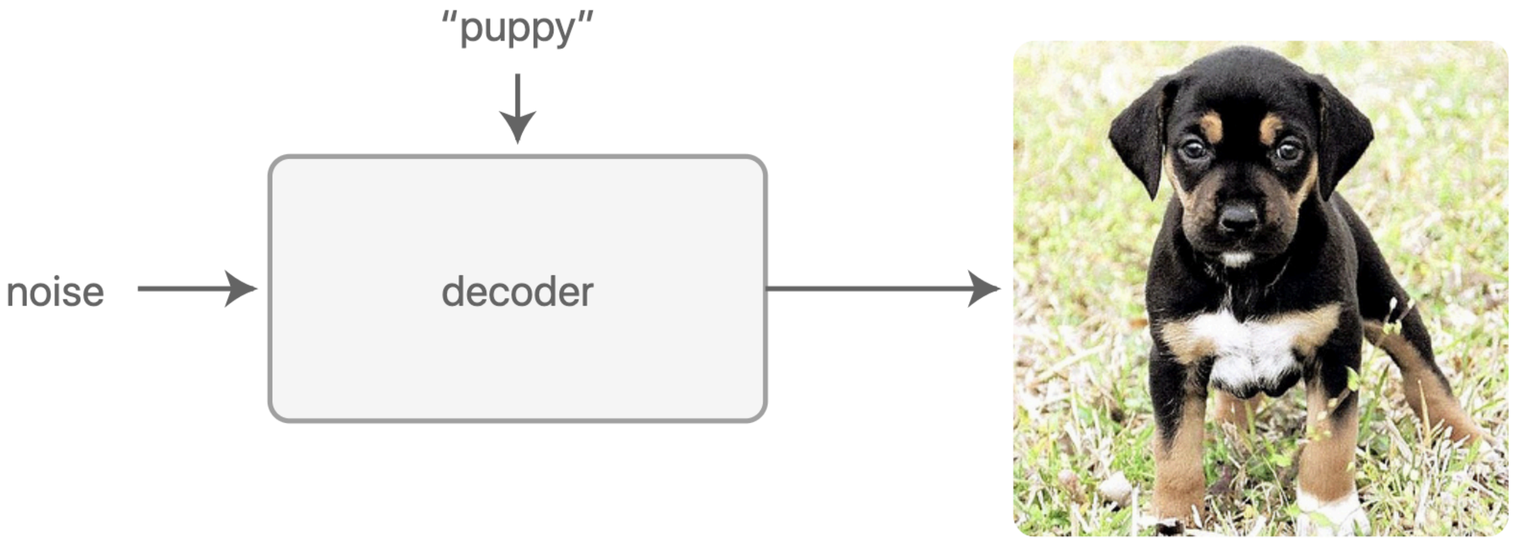




https://distill.pub/2018/feature-wise-transformations/

https://unsplash.com/photos/Lq6rcifGjOU

Conditioning

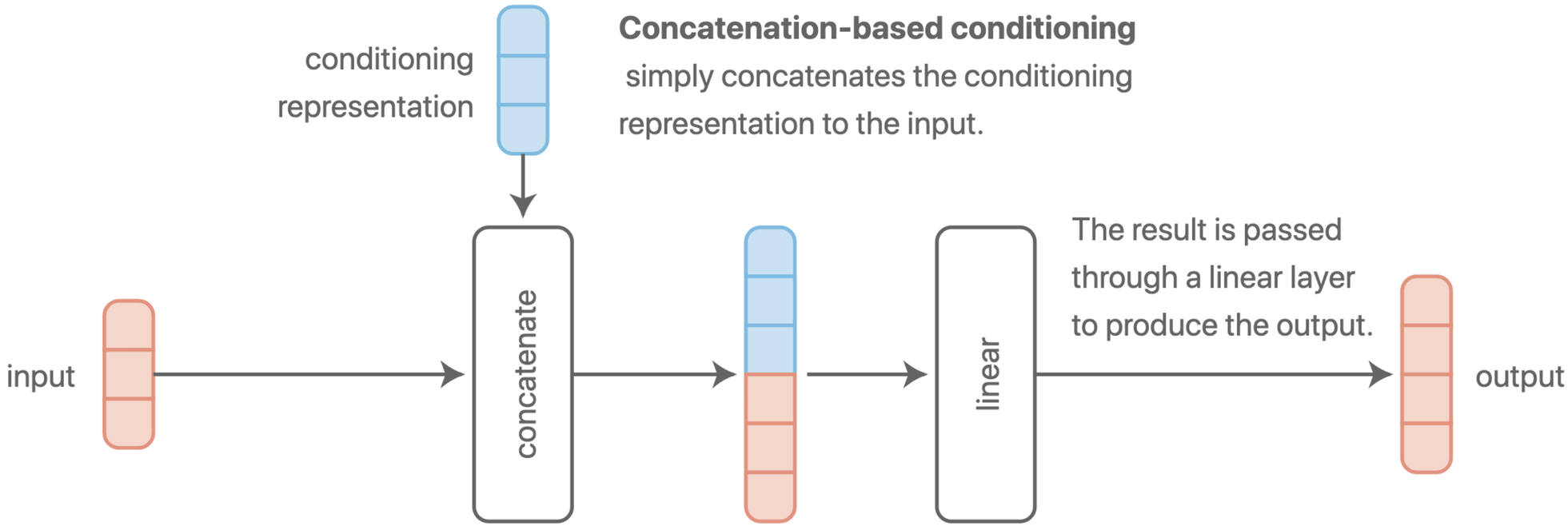


A **decoder-based generative model** maps a source of noise to a sample in the context of the "puppy" class.

- What if we have 1000 classes?

https://distill.pub/2018/feature-wise-transformations/

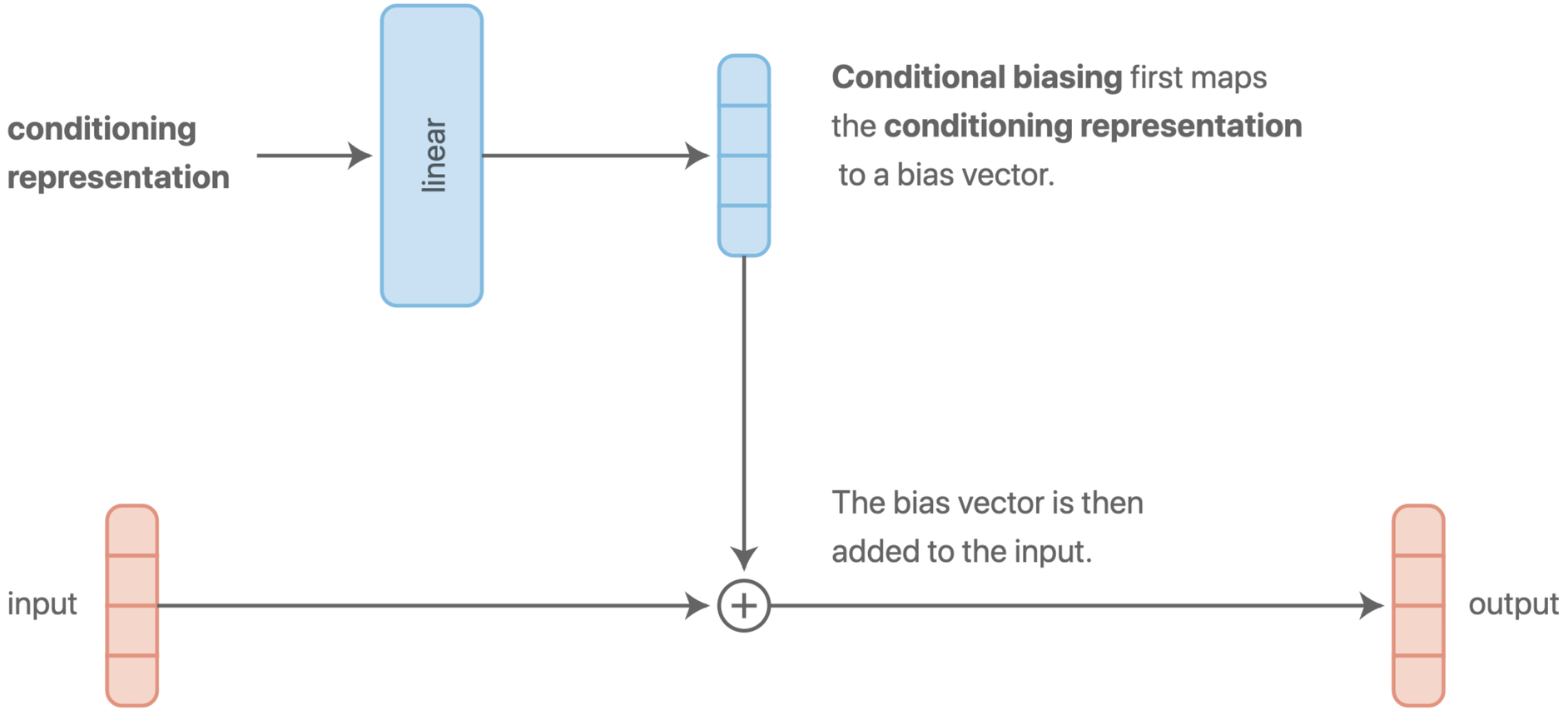
Concatenation-based conditioning



- implicit approach
- feature aggregation / detection

https://distill.pub/2018/feature-wise-transformations/

Conditional biasing



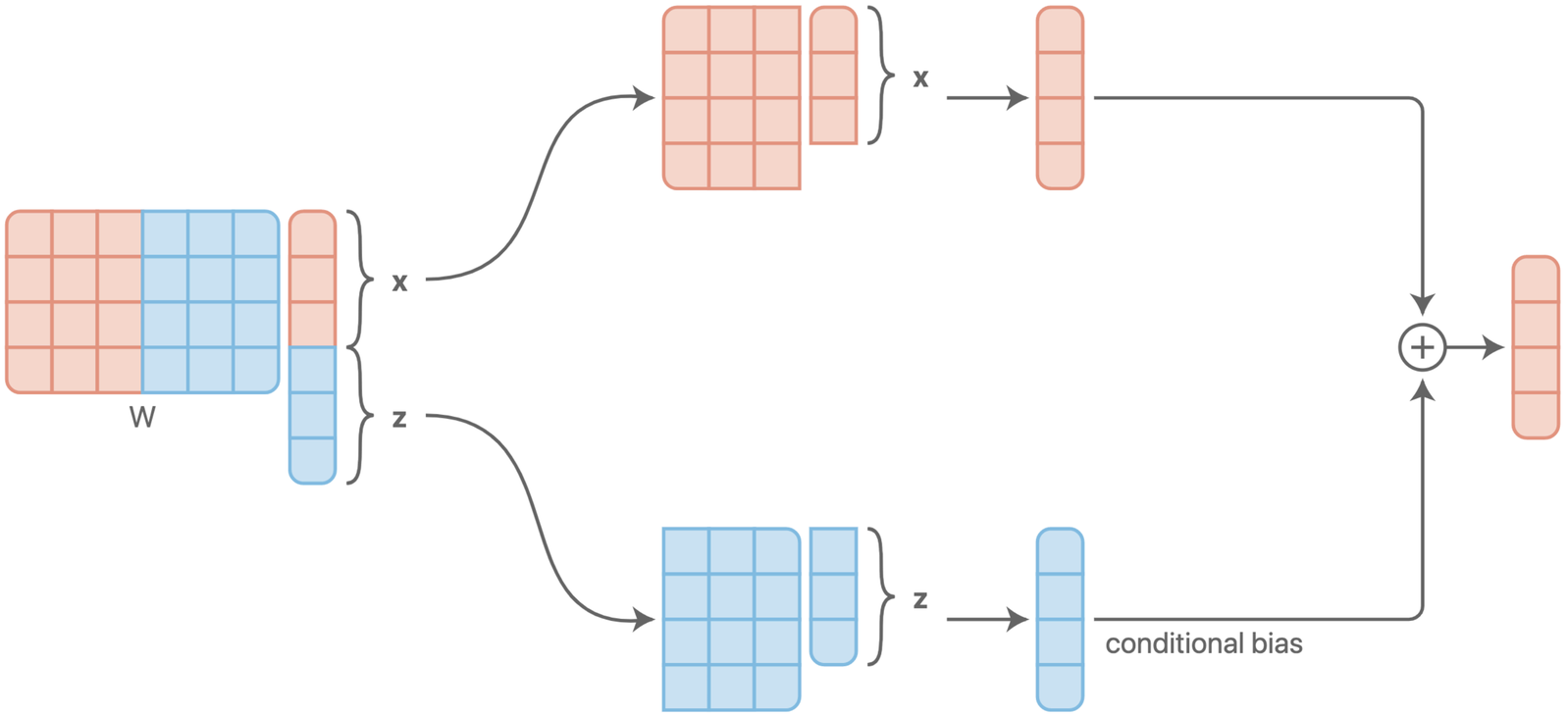
https://distill.pub/2018/feature-wise-transformations/

Concatenation == Biasing

Concatenation-based conditioning is equivalent to **conditional biasing**.

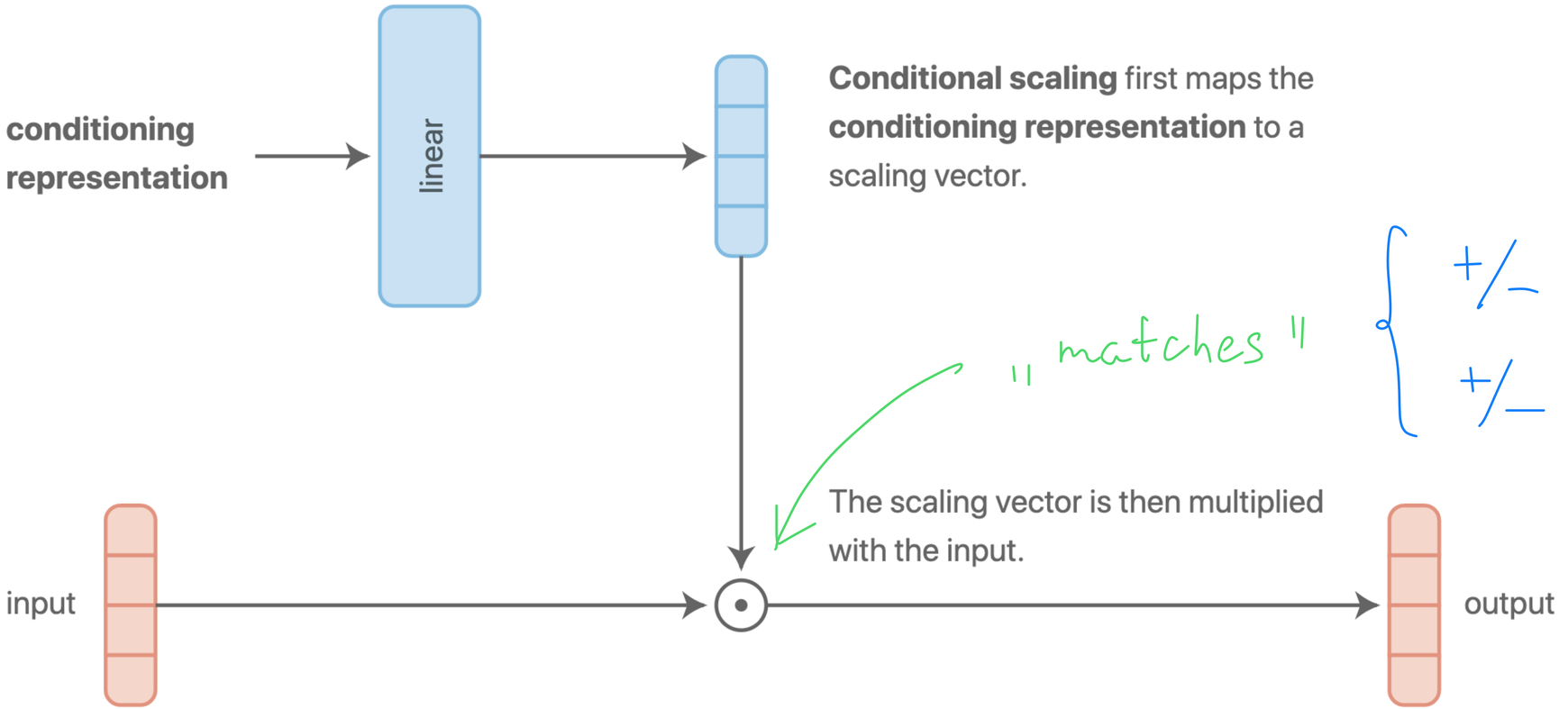
We can decompose the matrix-vector product into two matrix-vector subproducts.

We can then add the resulting two vectors. The **z**-dependent vector is a conditional bias.



https://distill.pub/2018/feature-wise-transformations/

Conditional scaling



Example: feature wise sigmoidal gating

https://distill.pub/2018/feature-wise-transformations/

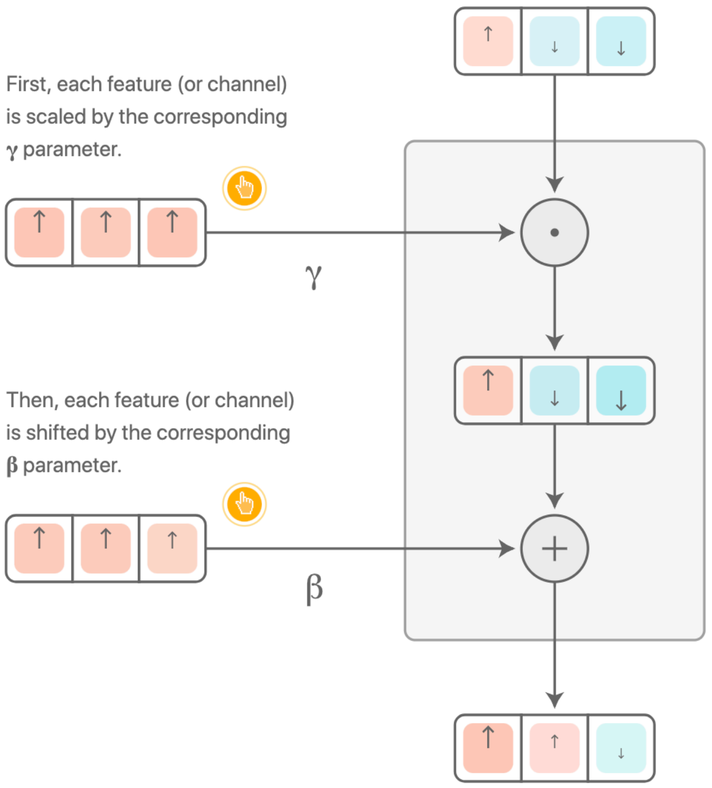
$$\mathbf{FiLM}(\mathbf{x}) = \gamma(\mathbf{z}) \odot \mathbf{x} + \beta(\mathbf{z})$$

- Scales linearly
  $O(N)$
  ↑ number of features
- Enough capacity for most scenarios

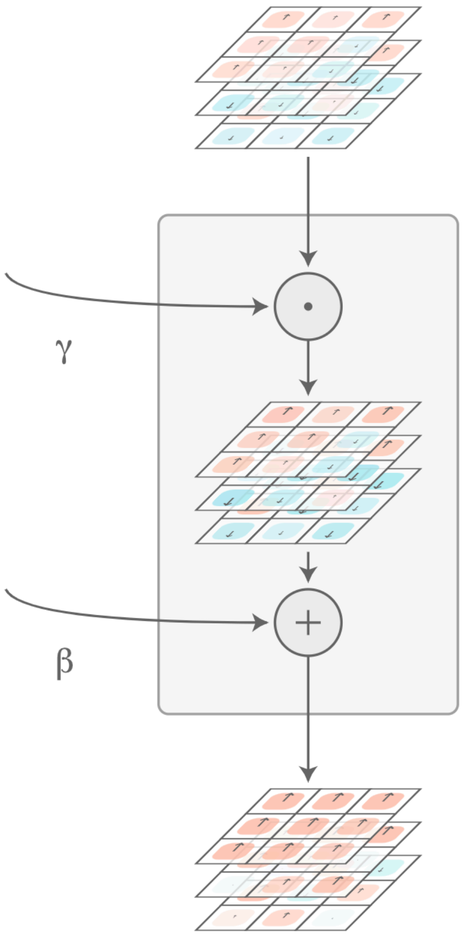
In a **fully-connected** network, FiLM applies a different affine transformation to each feature.

First, each feature (or channel) is scaled by the corresponding γ parameter.

Then, each feature (or channel) is shifted by the corresponding β parameter.



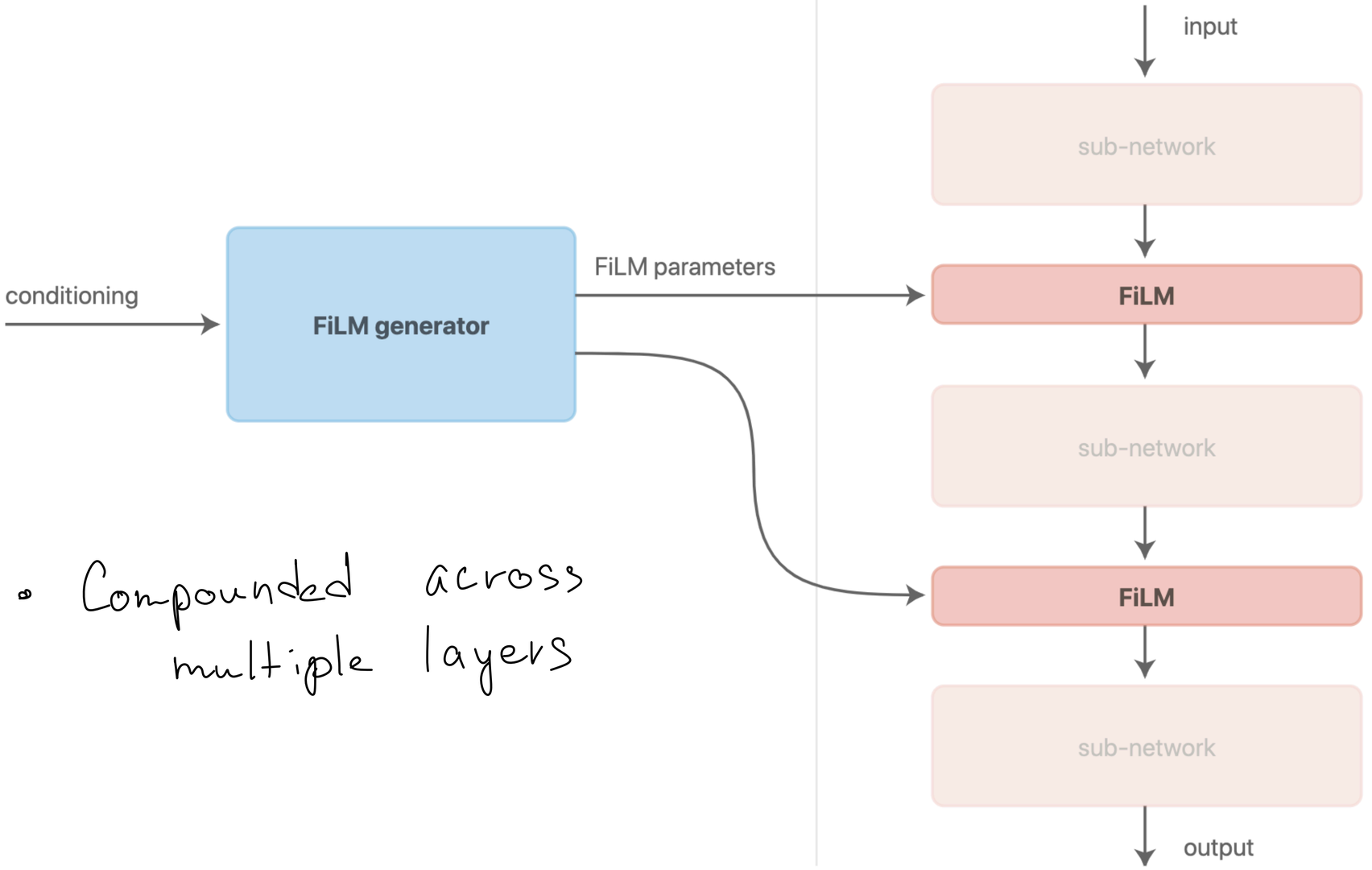
In a **convolutional** network, FiLM applies a different affine transformation to each channel, consistent across spatial locations.



https://distill.pub/2018/feature-wise-transformations/

The **FiLM generator** processes the conditioning information and produces parameters that describe how the target network should alter its computation.

Here, the **FiLM-ed network**'s computation is conditioned by two FiLM layers.



- Compounded across multiple layers

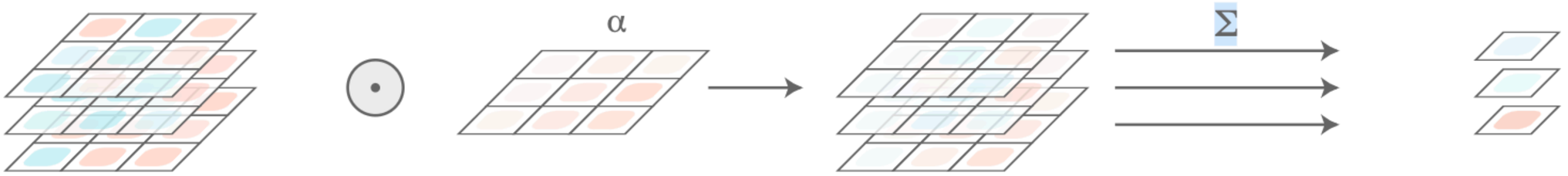
https://distill.pub/2018/feature-wise-transformations/


Attention vs FiLM

Attention computes a probability distribution over **locations**.

Attention pools over locations.

Attention summarizes the input into a vector.



„Specific locations / time-steps contain useful information"

FiLM computes a scaling vector applied to the **feature** axis.

FiLM conserves input dimensions.

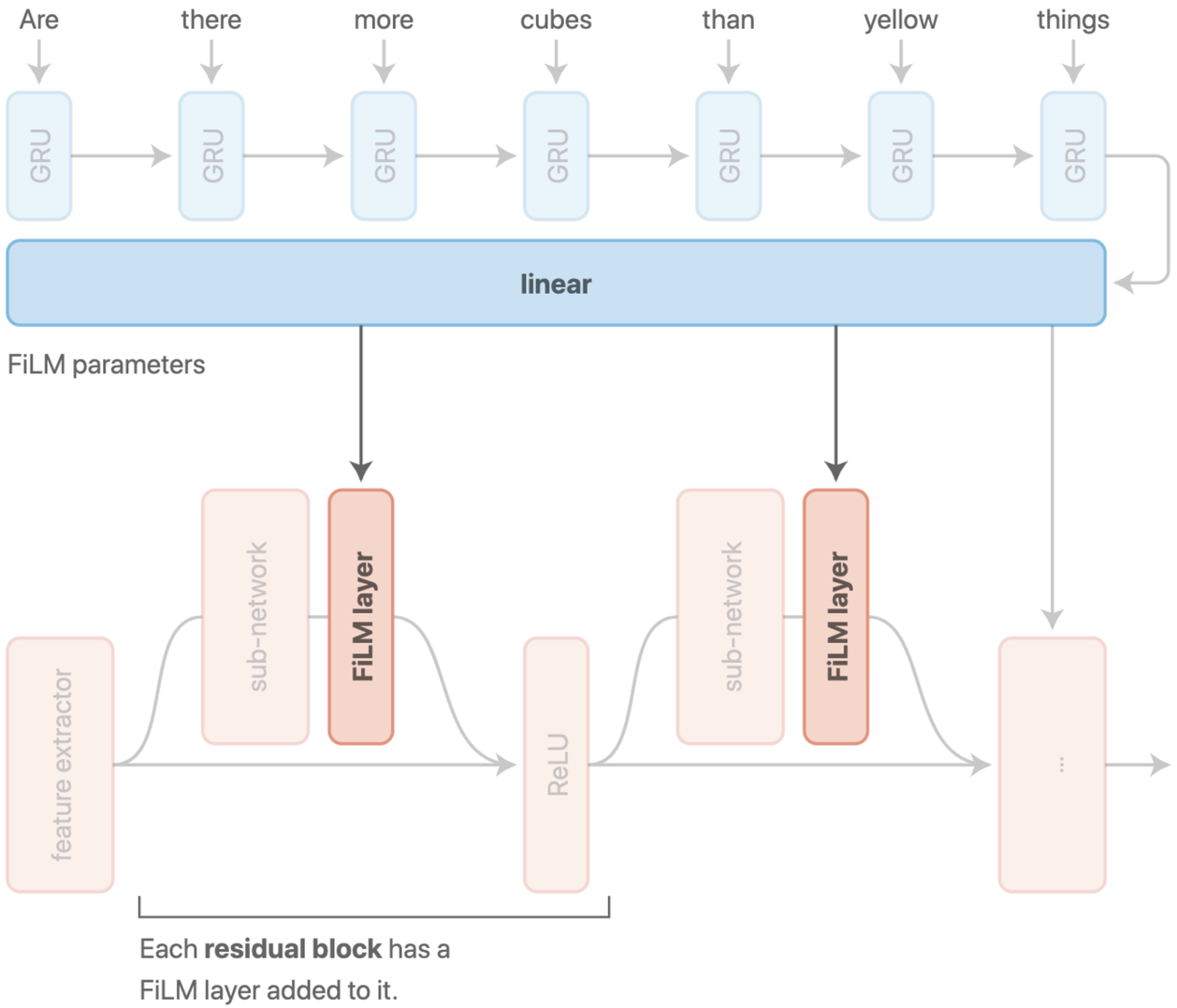
„Specific features contain useful information"

https://distill.pub/2018/feature-wise-transformations/

VQA

The **linguistic pipeline** acts as the FiLM generator.

FiLM layers in each residual block modulate the **visual pipeline**.



Learning visual reasoning without strong priors [PDF]
E. Perez, H. de Vries, F. Strub, V. Dumoulin, A. Courville.
ICML Workshop on Machine Learning in Speech and Language Processing. 2017.

FiLM: Visual Reasoning with a General Conditioning Layer [PDF]
E. Perez, F. Strub, H.d. Vries, V. Dumoulin, A. Courville.
AAAI. 2018.

https://distill.pub/2018/feature-wise-transformations/

Style transfer

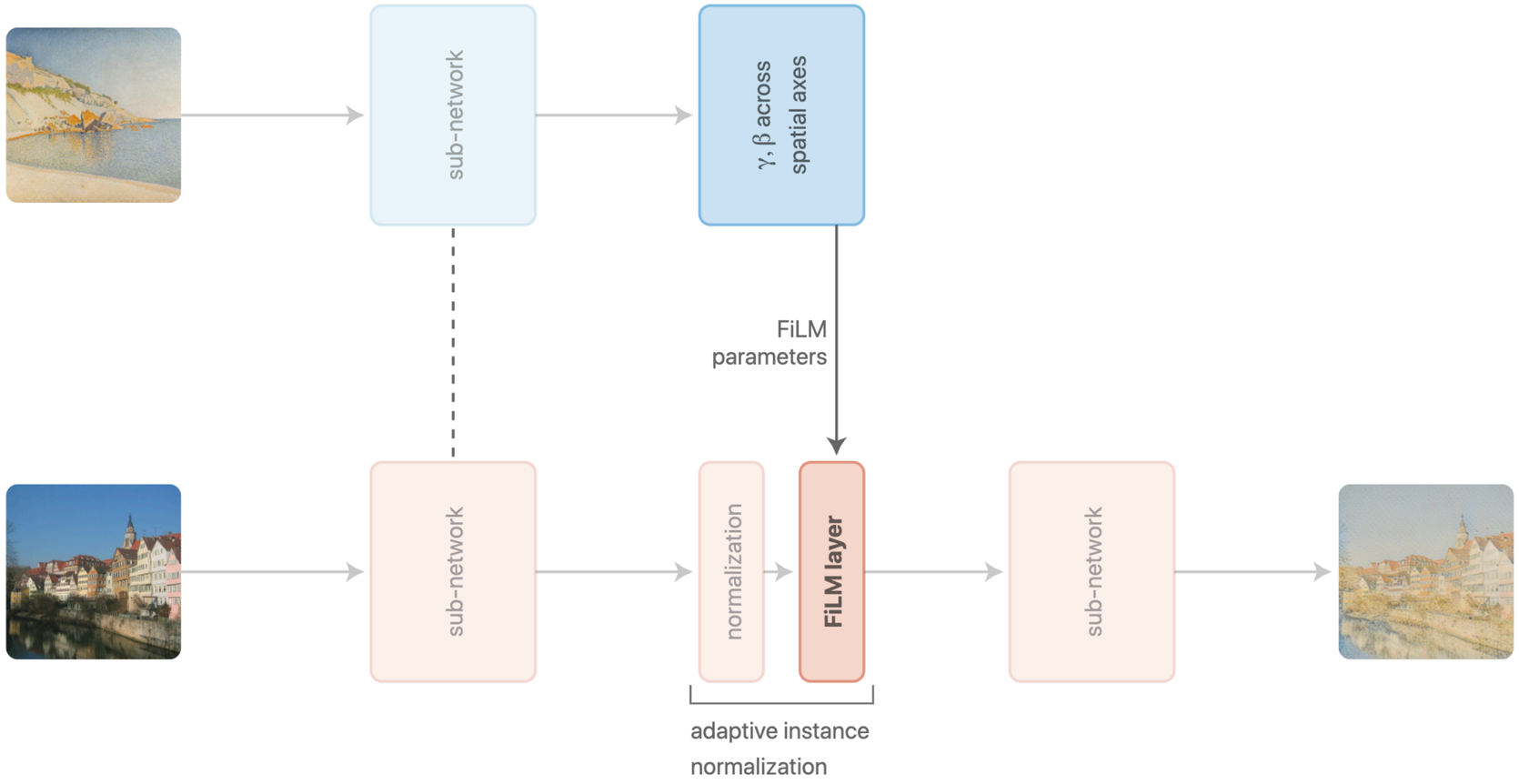
Conditional Batch Normalization

$$z = \gamma_s \left(\frac{x - \mu}{\sigma} \right) + \beta_s$$

The model processes content and style images up to the adaptive instance normalization layer.

FiLM parameters are computed as the spatial mean and standard deviation statistics of the style feature maps.

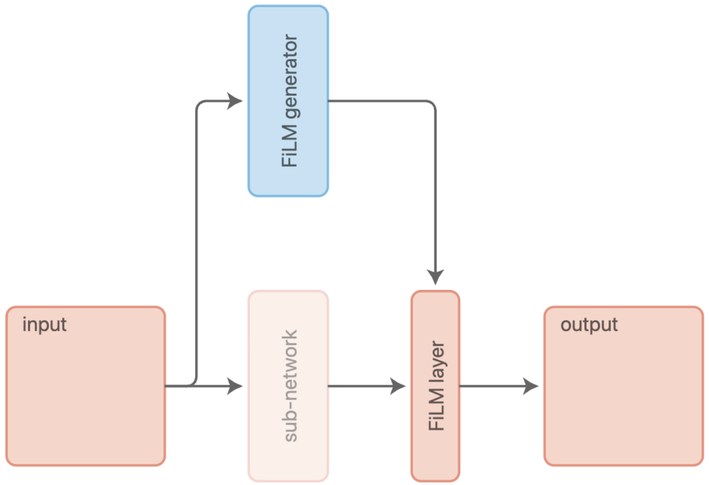
The FiLM-ed feature maps are fed to the remainder of the network to produce the stylized image.

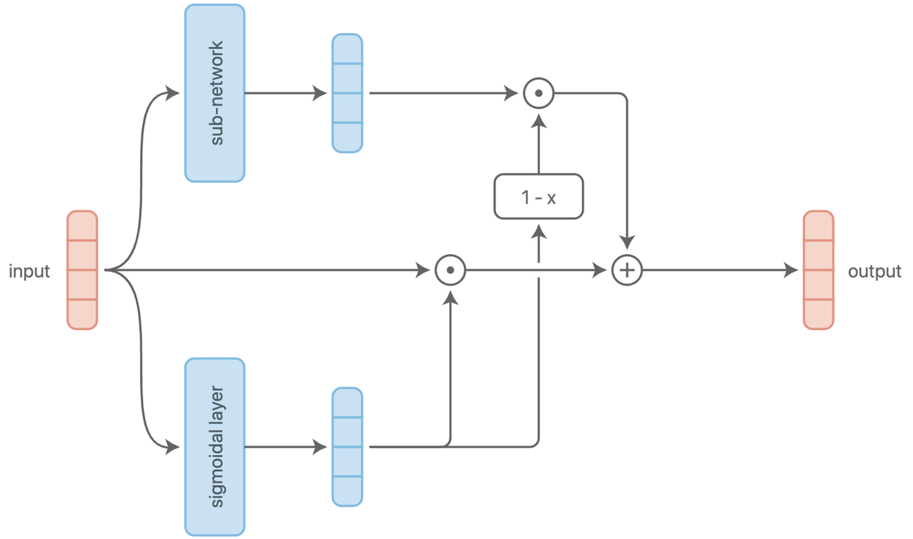


Arbitrary style transfer in real-time with adaptive instance normalization [PDF]
X. Huang, S. Belongie.
Proceedings of the International Conference on Computer Vision. 2017.

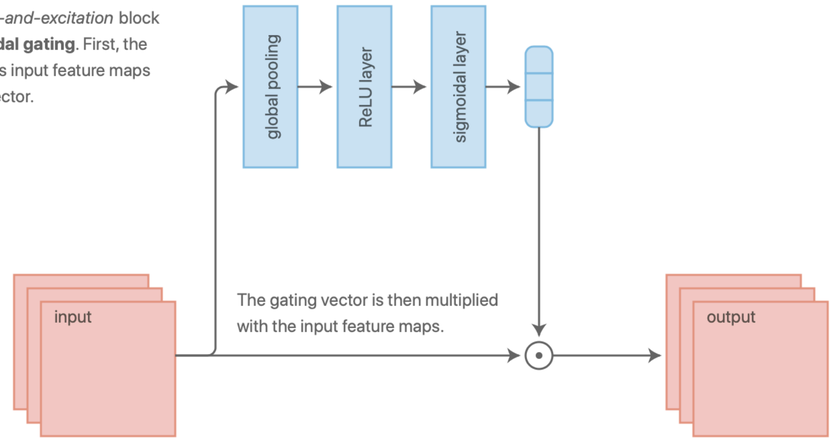
https://distill.pub/2018/feature-wise-transformations/

Image recognition: Highway Networks and Squeeze-and-excitation





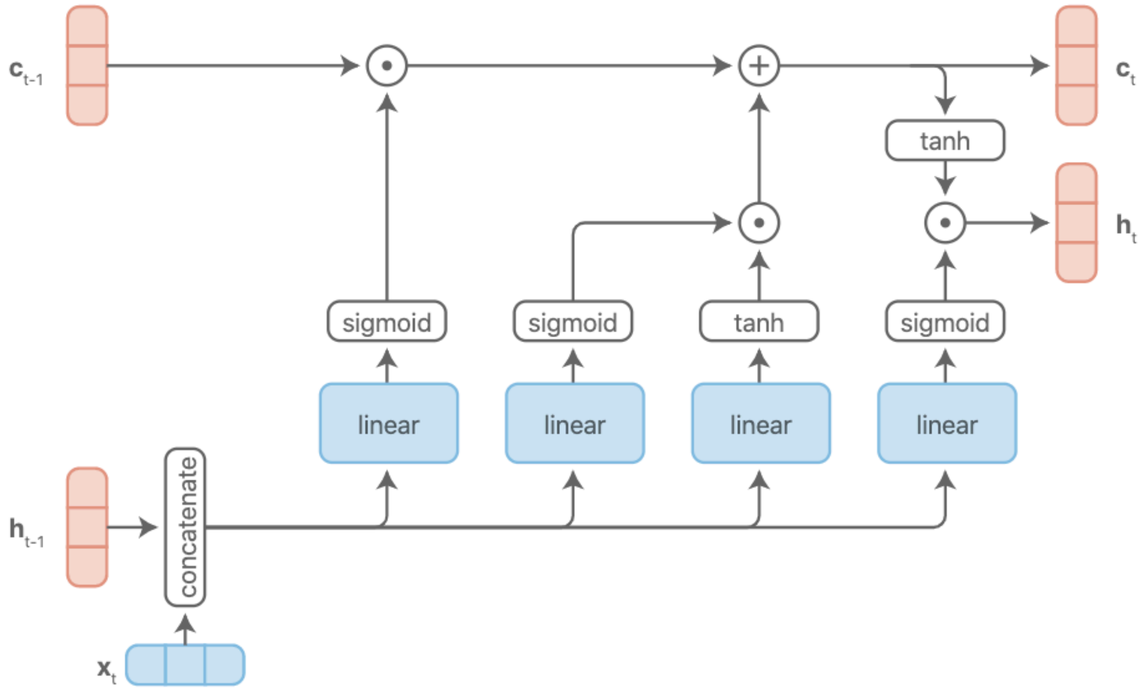
The *squeeze-and-excitation* block uses **sigmoidal gating**. First, the network maps input feature maps to a gating vector.

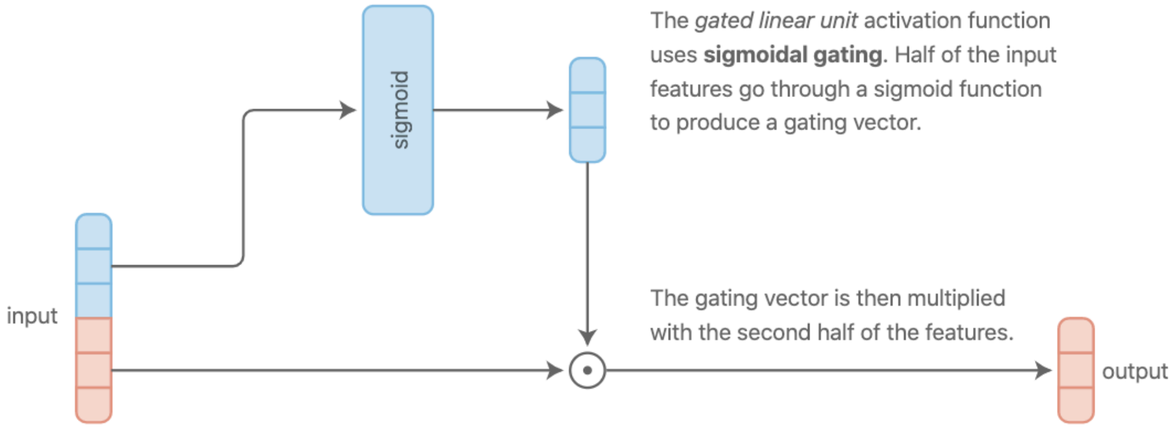


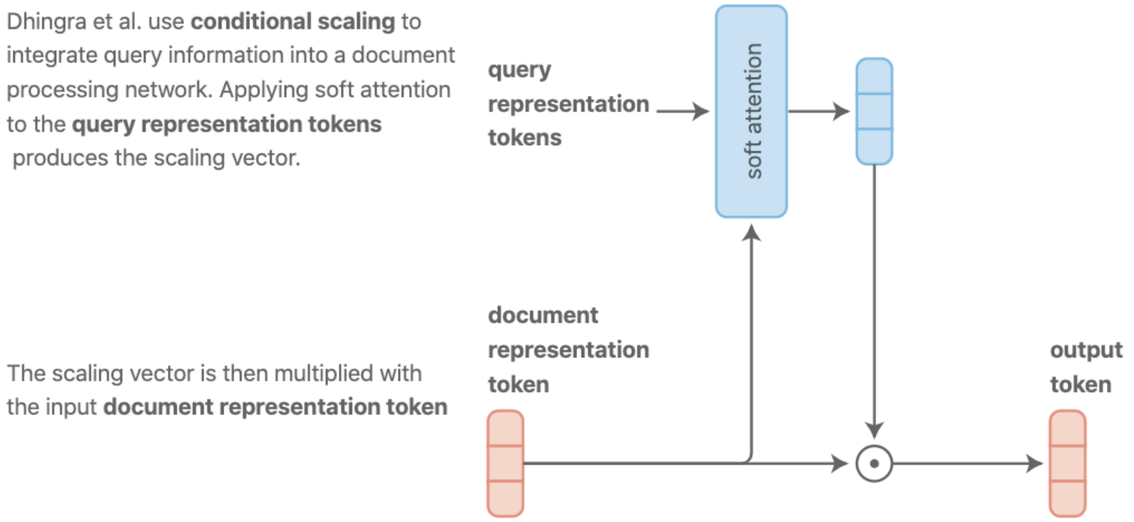
Highway networks [PDF]
R.K. Srivastava, K. Greff, J. Schmidhuber.
ICML Deep Learning Workshop. 2015.

https://distill.pub/2018/feature-wise-transformations/

NLP: LSTM, GLU and Gated Attention Reader





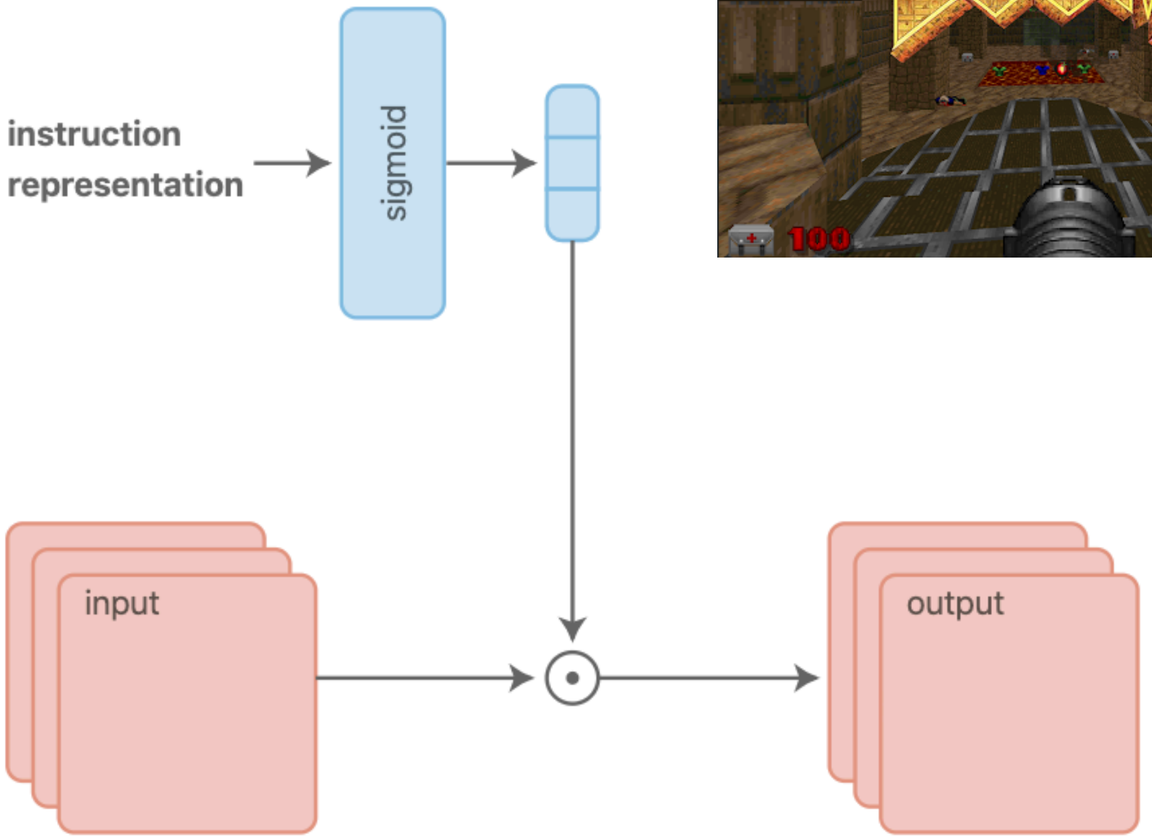


Long short-term memory [link]
S. Hochreiter, J. Schmidhuber.
Neural Computation, Vol 9(8), pp. 1735--1780. MIT Press. 1997.

https://distill.pub/2018/feature-wise-transformations/

RL

Chaplot et al. use **sigmoidal gating** as a multimodal fusion mechanism. An instruction representation is mapped to a scaling vector via a sigmoid layer.



The scaling vector is then multiplied with the input feature maps. A policy network uses the result to decide the next action.

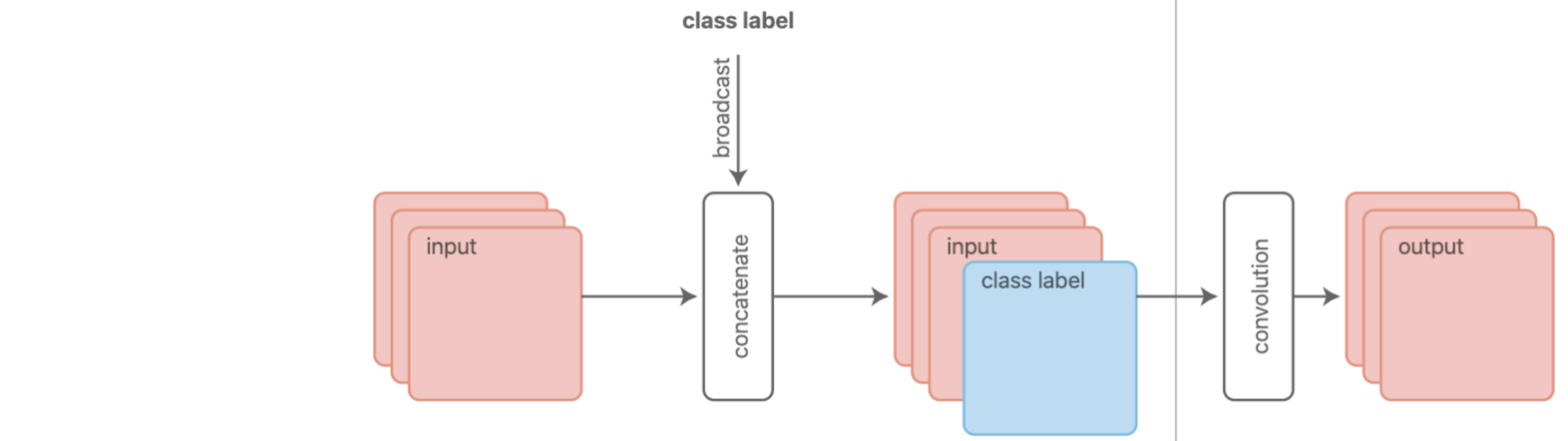
Gated-attention architectures for task-oriented language grounding [PDF]
D.S. Chaplot, K.M. Sathyendra, R.K. Pasumarthi, D. Rajagopal, R. Salakhutdinov.
ACL Workshop on Language Grounding for Robotics. 2017.

https://distill.pub/2018/feature-wise-transformations/

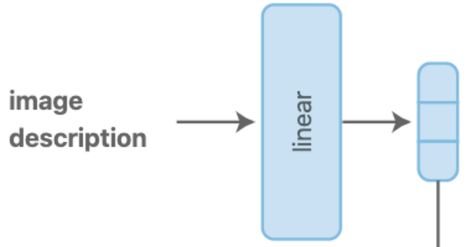
Generative modeling: DCGAN, PixelCNN

Concatenation-based conditioning is used in the class-conditional DCGAN model. Each convolutional layer is concatenated with the broadcased label along the channel axis.

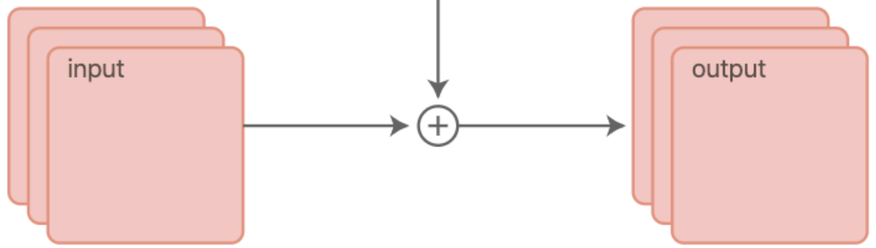
The resulting stack of feature maps is then **convolved** to produce the conditioned output.



PixelCNN uses **conditional biasing**. The model first maps a high-level image description to a bias vector.



Then, it adds the bias vector to the input stack of feature maps to condition convolutional layers.

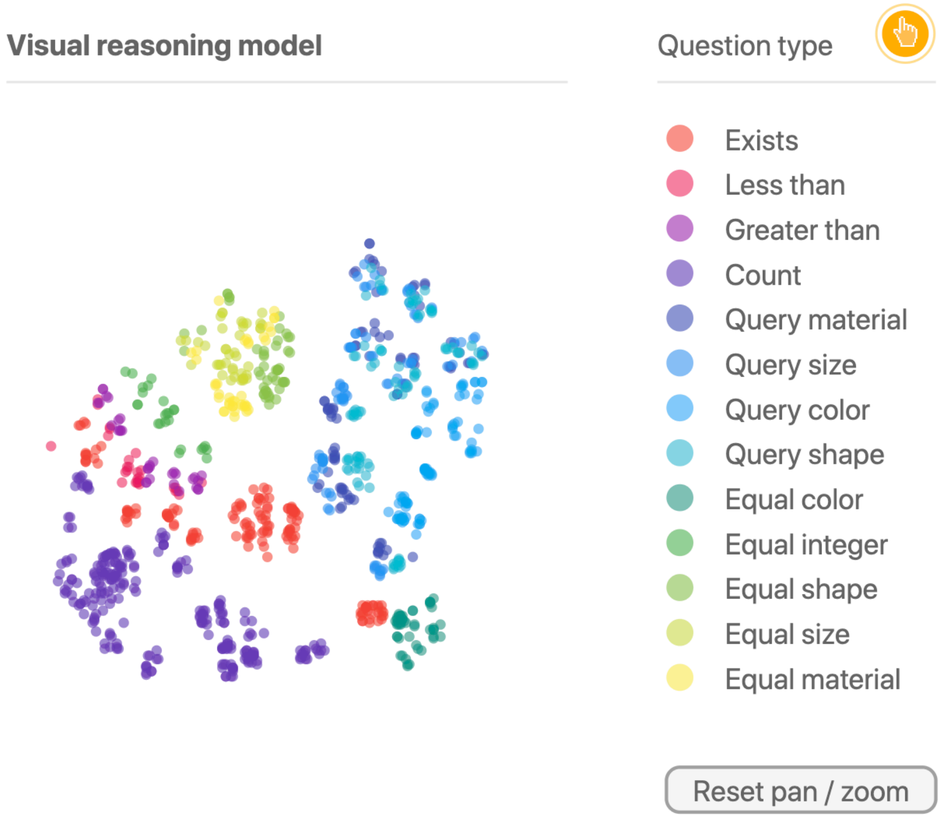


https://distill.pub/2018/feature-wise-transformations/

FiLM as a task representation

$$\mathrm{FiLM}(\mathbf{x}) = \gamma(\mathbf{z}) \odot \mathbf{x} + \beta(\mathbf{z})$$

- each task
  - question about an image
  - a painting style to imitate
  - elicits a different set of FiLM parameters via the FiLM generator
    - representation in terms of how to modulate the FiLM-ed network

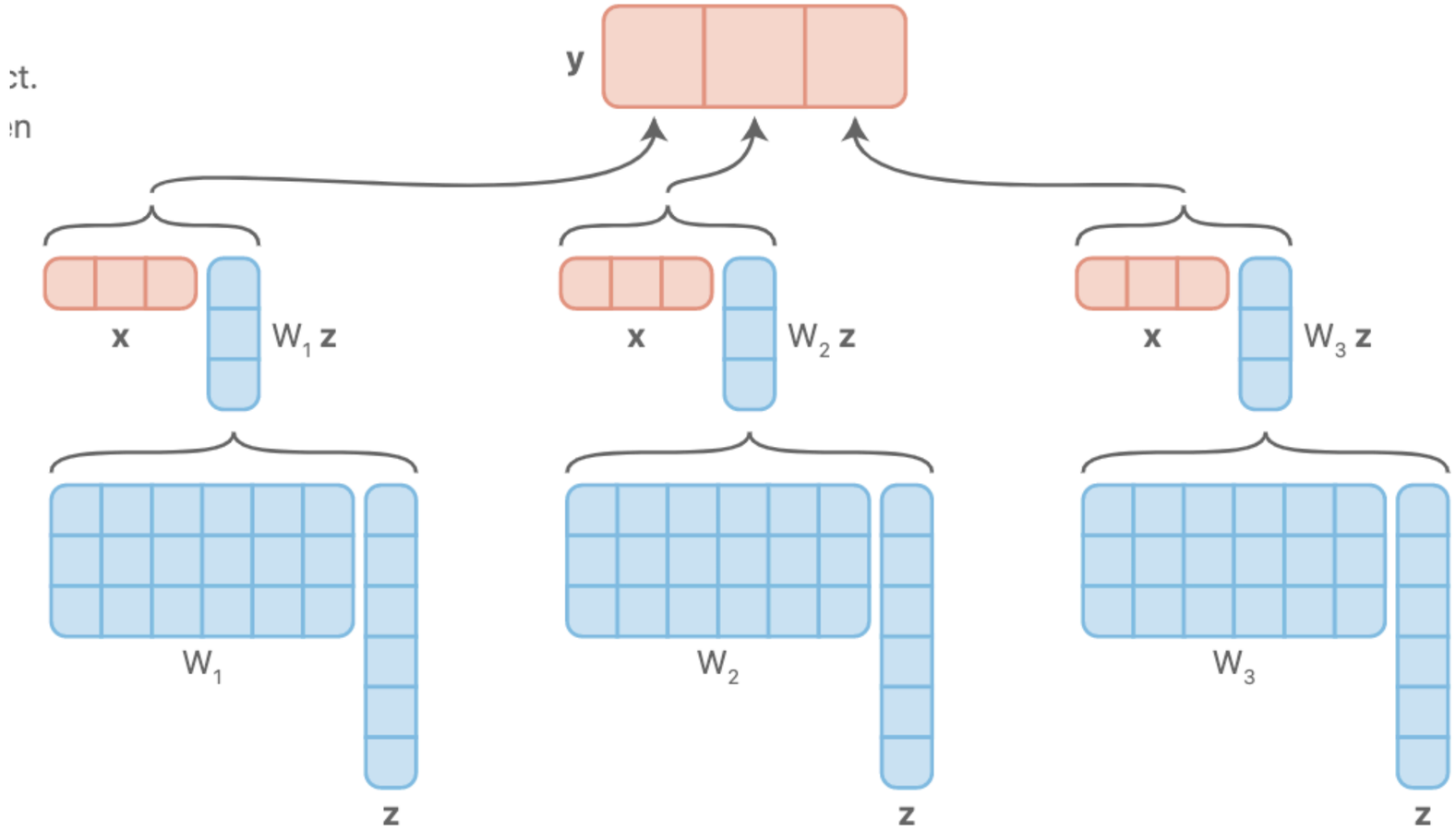


Billinear form

https://unsplash.com/photos/MyU3s1VasA4

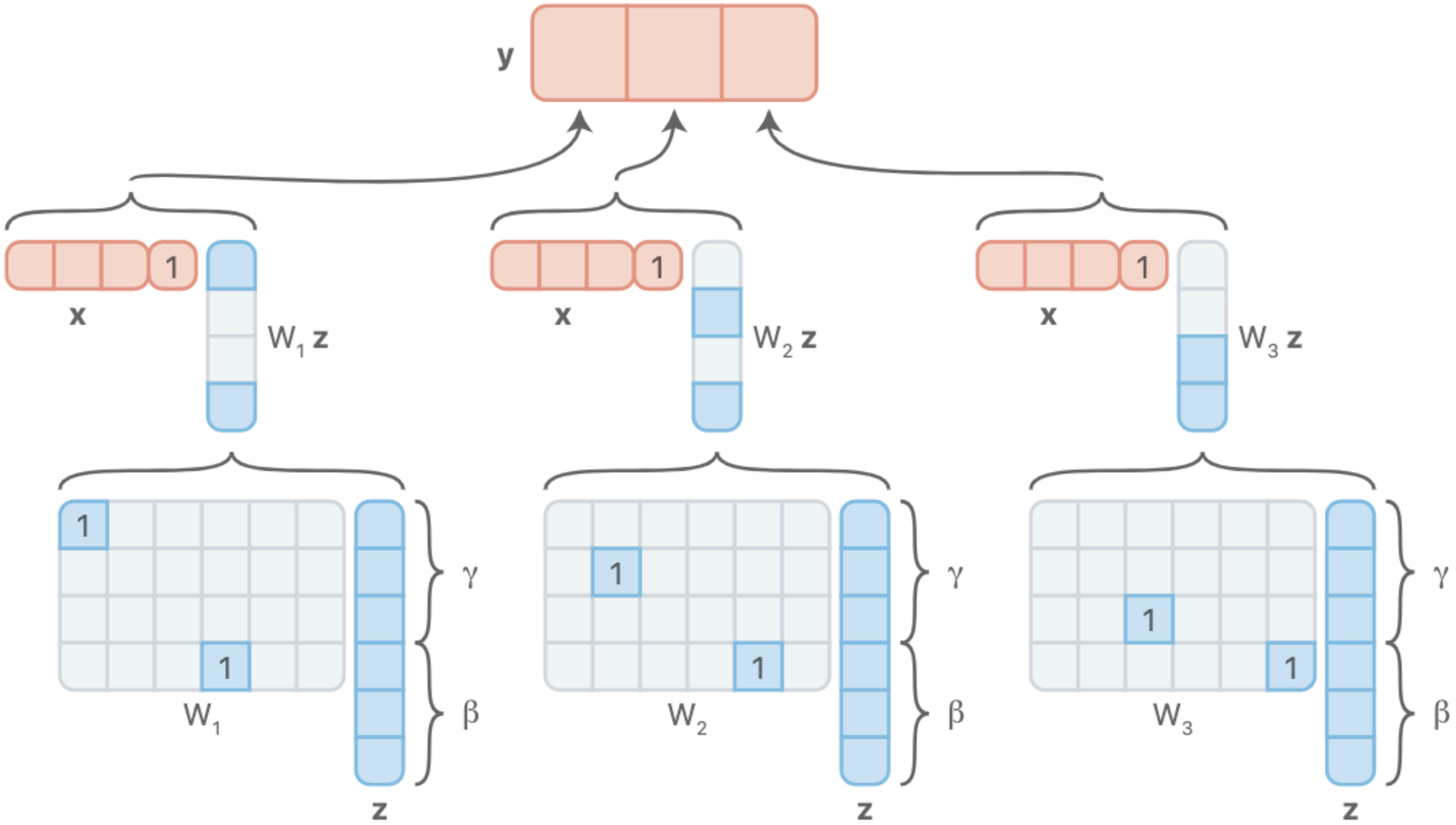
Bilinear form

$$y_k = \mathbf{x}^T W_k \mathbf{z}$$



https://distill.pub/2018/feature-wise-transformations/

FiLM: Low-Rank Bilinear form



https://distill.pub/2018/feature-wise-transformations/

Multiplicative Interactions

Multiplicative Interactions and Where to Find Them https://openreview.net/pdf?id=rylnK6VtDH

Additive form

Given arbitrary inputs **x** and **z**, where we are trying to model **f(x, z)**, a single layer might typically look like:

$$f(\mathbf{x}, \mathbf{z}) = \mathbf{W}[\mathbf{x}; \mathbf{z}] + \mathbf{b}$$

where $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{z} \in \mathbb{R}^m$, $\mathbf{W} \in \mathbb{R}^{(m+n)\times k}$

https://iclr.cc/virtual_2020/poster_rylnK6VtDH.html

Multiplicative Interactions

Instead, we now have the **bilinear** form:

$$f(\mathbf{x}, \mathbf{z}) = \mathbf{z}^T \mathbb{W} \mathbf{x} + \mathbf{z}^T \mathbf{U} + \mathbf{V}\mathbf{x} + \mathbf{b}$$

$$\mathbf{x} \in \mathbb{R}^n, \mathbf{z} \in \mathbb{R}^m, \mathbb{W} \in \mathbb{R}^{m \times n \times k} \text{ and } (\mathbf{z}^T \mathbb{W} \mathbf{x})_k = \sum_{ij} \mathbf{z}_i \mathbb{W}_{ijk} \mathbf{x}_j.$$

https://iclr.cc/virtual_2020/poster_rylnK6VtDH.html

HyperNetworks

$$f(\mathbf{x}, \mathbf{z}) = \mathbf{z}^T \mathbb{W} \mathbf{x} + \mathbf{z}^T \mathbf{U} + \mathbf{V}\mathbf{x} + \mathbf{b}$$

Set $\boxed{\mathbf{W}' = \mathbf{z}^T \mathbb{W} + \mathbf{V}}$ and $\boxed{\mathbf{b}' = \mathbf{z}^T \mathbf{U} + \mathbf{b}}$

$$f(\mathbf{x}, \mathbf{z}) = \mathbf{W}'\mathbf{x} + \mathbf{b}'$$

https://iclr.cc/virtual_2020/poster_rylnK6VtDH.html

Implementation

```
# Simple python code for MI Layers
import sonnet as snt
import tensorflow as tf

# A standard linear layer
# B is the batch size
# E is the input size
# C is the context size
x = ... # input of size [B, E]
z = ... # context of size [B, C]
xz = tf.concat([x,z], 1)
y = snt.Linear(output_size)(xz)
```

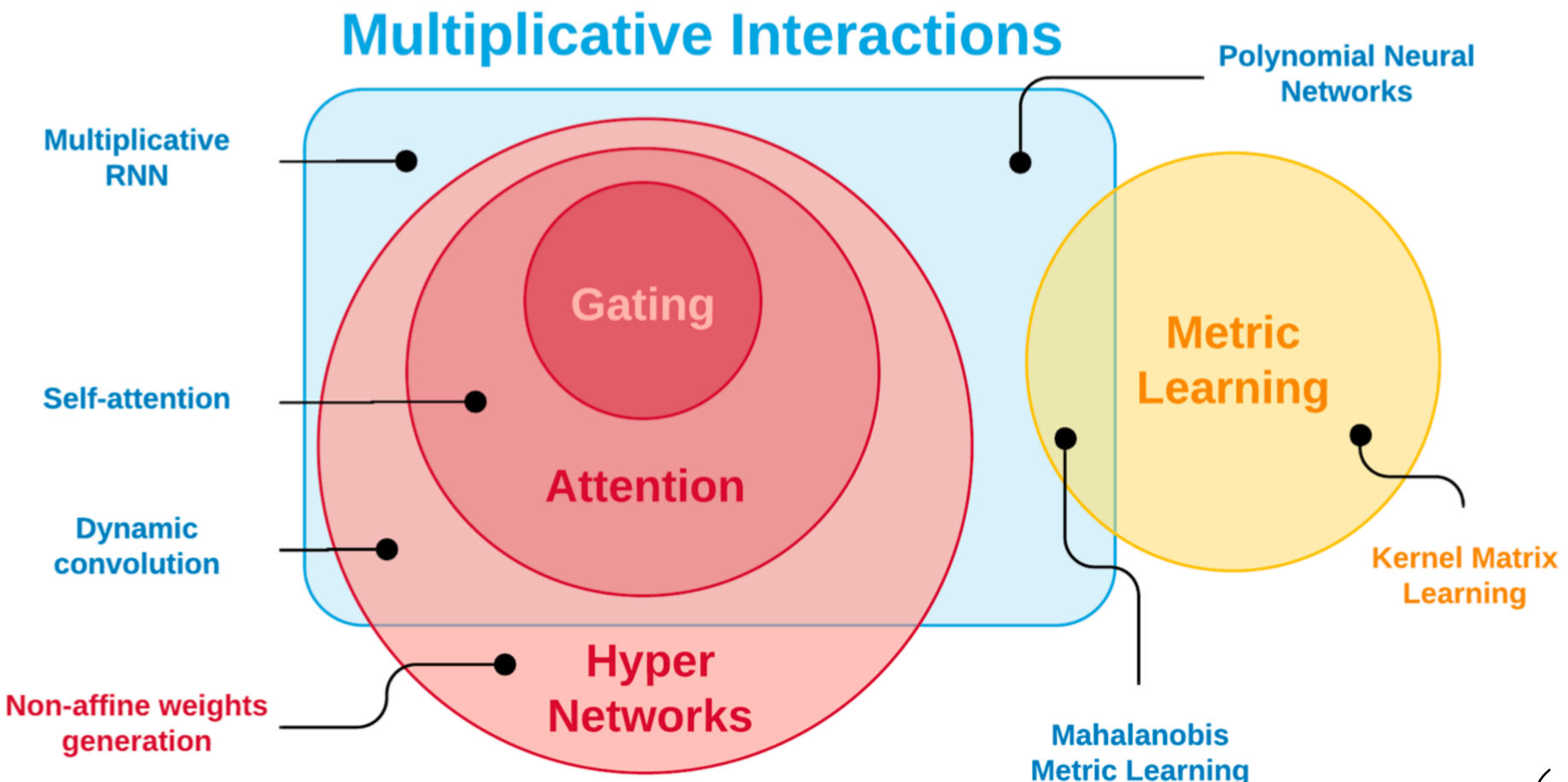
```
# Instead, we generate a W and b
# This defines an implicit 3D weight tensor

W = snt.Linear(output_size * input_size)(z)
b = snt.Linear(output_size)(z)

# Reshape to the correct shape
# Note: we have B weight matrices
# i.e. one per batch element
W = tf.reshape(W, [B, input_size, output_size])

# Output
y = tf.matmul(x, W) + b
```

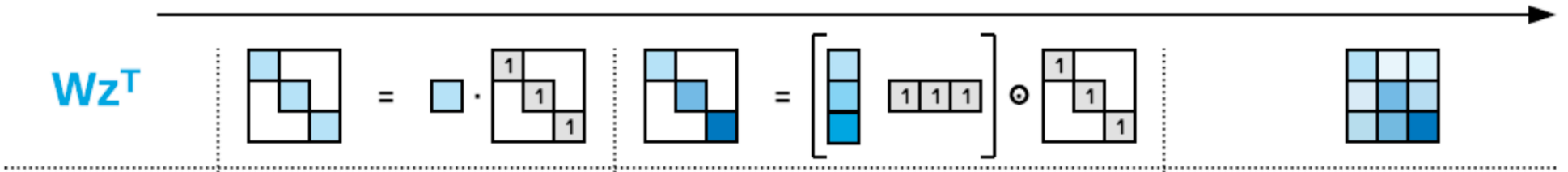
Multiplicative Interactions

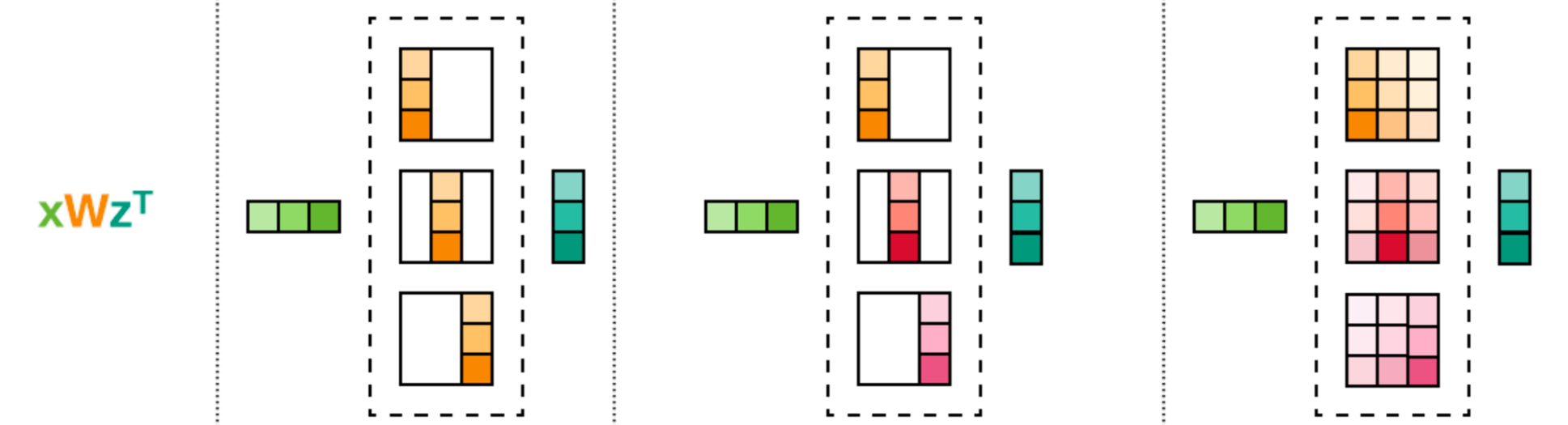


$$s\left(\frac{QK^T}{\sqrt{n}}\right)V$$

$$d_C(x,z) = \|x-z\|_C = (x-z)^T C^{-1}(x-z) =$$
$$= x^T C^{-1} x - 2x^T C^{-1} z + z^T C^{-1} z$$

Order of multiplicative interactions





Multiplicative interaction	Scaling	Hadamard product	General bilinear form
Projection matrix class	Scalar	Diagonal	Unconstrained
HyperNetwork output	Scalar	Vector	Matrix

Expressivity of the models

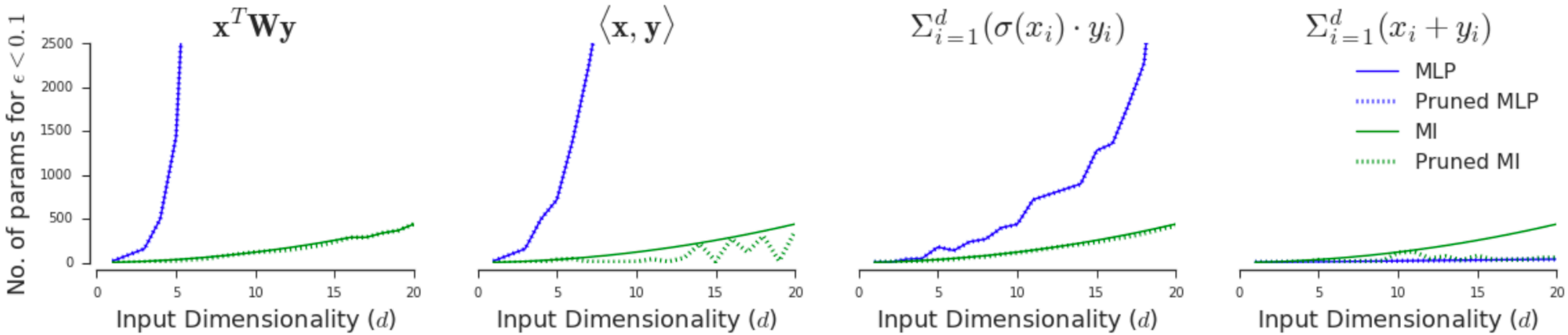


Figure 2: Number of parameters needed for a regular, single layer MLP (blue line) to represent the function up to 0.1 MSE over the domain of a standard d-dimensional Gaussian compared to the same quantity for a multiplicative model (green line). σ denotes sigmoid. Dotted lines represent pruned models where all weights below absolute value of 0.001 were dropped. Note that for MLP all parameters are actually used, while for MI module some of these functions (summation and dot product) can be compactly represented with pruning.

Toy Multi-task Learning

$$y = a_i x + b_i \text{ and } y = a_i \sin(10x) + b_i$$

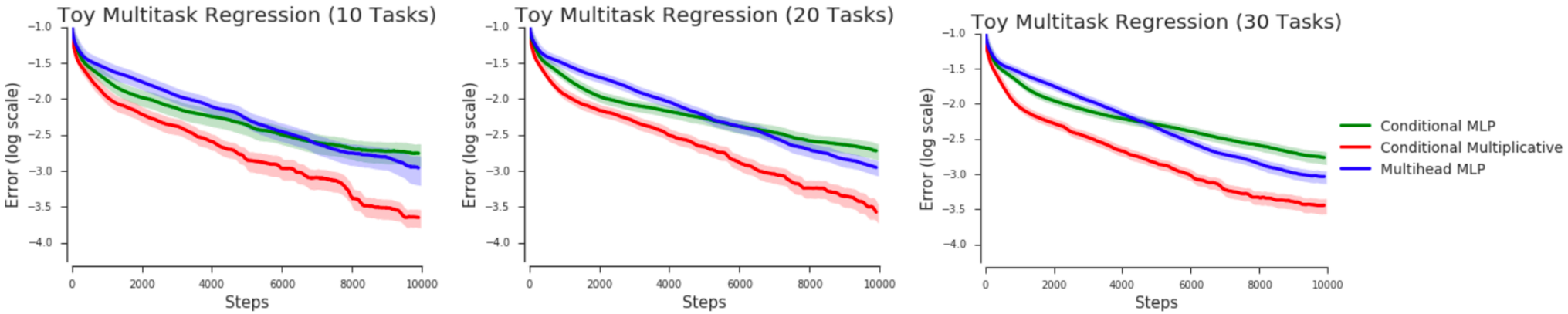


Figure 3: Averaged learning curves for different models while varying the number of tasks in the toy multitask regression domain. Shaded regions represent standard error of mean estimation.

Multi-Task RL on DMLab-30

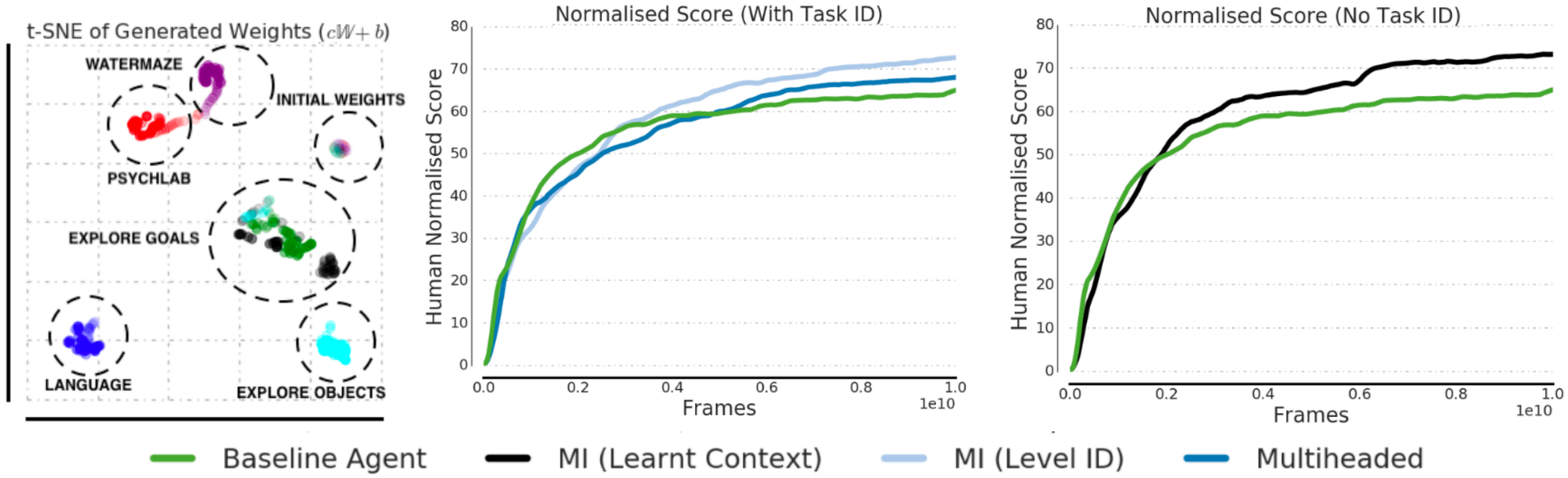


Figure 4: (a) A t-SNE plot of the generated weights from an $\mathcal{M}$ layer. (b) Human normalised performance (capped at 100) when using task ID as context to an $\mathcal{M}$ layer. (c) Using a learnt context instead.

Language Modeling

Table 1: Word-level perplexity on WikiText-103

Model	Valid	Test	No. Params
LSTM Rae et al. (2018)	34.1	34.3	88M
Gated CNN Dauphin et al. (2017)	-	37.2	-
RMC Santoro et al. (2018)	30.8	31.6	-
Trellis Networks Bai et al. (2019)	-	30.35	180M
TransformerXL Dai et al. (2018)	**17.7**	**18.3**	**257M**
LSTM (ours)	34.7	36.7	88M
LSTM + MultDec	31.7	33.7	105M
LSTM + MultEncDec	**28.9**	**30.3**	**110M**


Thank you