FiLM: A Simple Way to Let One Input Guide Another

Fengnan Deng

August 2026

The idea in one sentence. FiLM lets context—such as a question, instruction, or style—amplify, suppress, or shift a neural network's intermediate features.

1 The same input can require different thinking

Imagine showing a model one image and asking two questions:

> "What color is the cube?"
> "How many cylinders are there?"

The image has not changed, but the useful information has. The first question asks the model to care about color and shape. The second asks it to care about object type and counting. A good model should not process the image in exactly the same way for both questions.

This is a *conditioning* problem. One input contains the main content, while another input tells the model what kind of computation is needed. Feature-wise Linear Modulation, usually shortened to **FiLM**, is a particularly simple way to connect the two [1].

The easiest analogy is an audio equalizer. The music is the main input. The equalizer settings do not replace the music; they change which parts sound stronger or weaker. FiLM does something similar inside a neural network. The context produces a set of small control knobs, and those knobs adjust the network's intermediate features.

2 Two paths, one prediction

A FiLM model has two paths:

- the **main path** processes the content, such as an image; and
- the **conditioning path** processes the context, such as a question.

The conditioning path does not normally produce the answer by itself. Instead, it produces two controls for each group of features: a *scale* and a *shift*. These controls are inserted into one or more blocks of the main network, as shown in Figure 1.

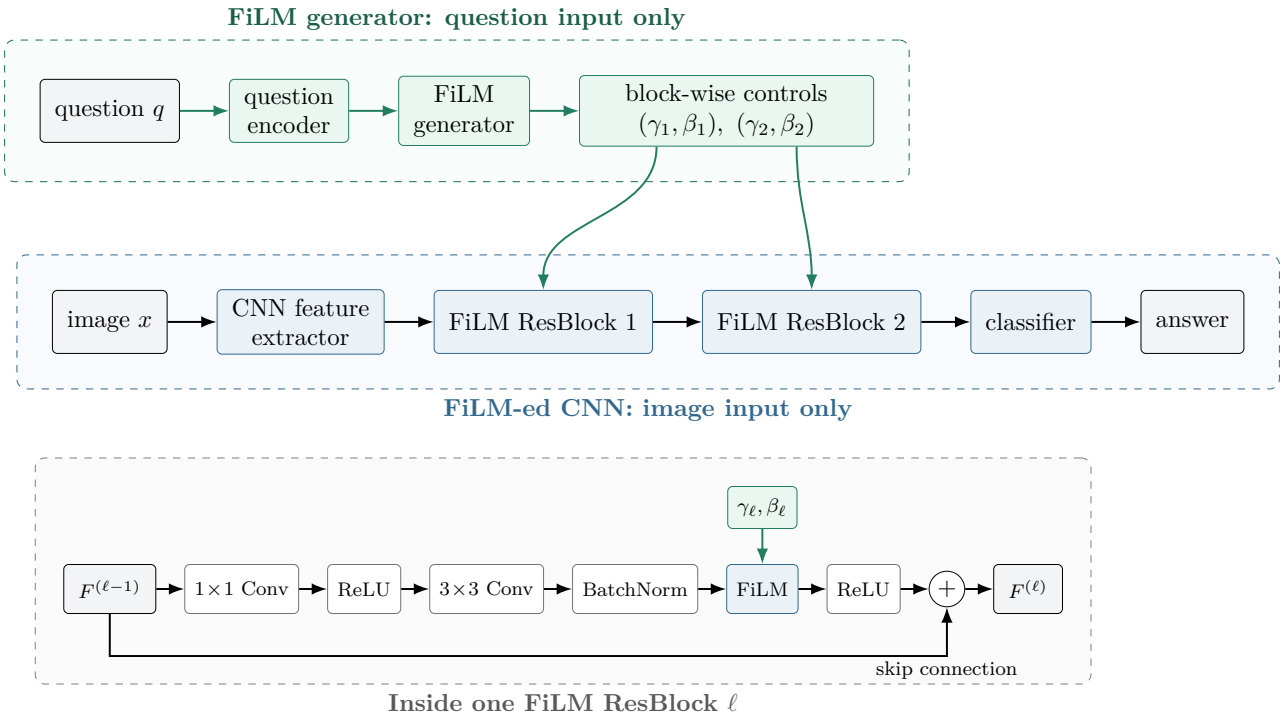

Figure 1: The question and image follow separate paths. For clarity, only two FiLM residual blocks are shown. The third row expands one FiLM residual block, including its convolutional branch, FiLM operation, and skip connection.

A shared CNN can answer questions about color, counting, or spatial relations because the conditioning path adjusts it for each question. A separate CNN for every kind of question is not needed. Inside block ℓ, the generated $(\gamma_\ell, \beta_\ell)$ values enter the FiLM operation after BatchNorm.

3 The whole operation in one line

The FiLM operation can be written in plain language as

$$\boxed{\text{new feature} = \text{scale from context} \times \text{old feature} + \text{shift from context}} \tag{1}$$

In the notation used in most papers, the same line is

$$\widetilde{F} = \gamma(q) \odot F + \beta(q). \tag{2}$$

There are only four pieces to remember:

- F is an existing feature inside the main network;
- q is the context, such as the question;
- $\gamma(q)$ is the scale predicted from that context; and
- $\beta(q)$ is the shift predicted from that context.

The symbol $\odot$ simply means that each feature gets its own scale. For a CNN, each feature channel receives its own scale and shift, and those values are shared across all spatial locations in that channel.

For example, scaling a color feature from 4 by 0.25 reduces it to 1, while scaling a counting feature by 1.5 makes it more prominent. The shift moves the baseline and can determine whether the next layer reacts. Together, these controls amplify useful features, suppress distractions, or switch features off.

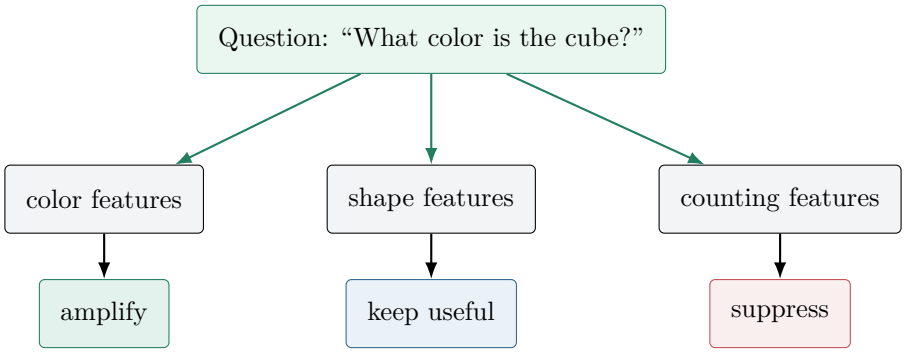

Figure 2: An intuitive example of feature selection. This is a conceptual picture, not a measured result: the learned controls decide which feature groups deserve more or less influence for the current question.

4 How does FiLM learn the right controls?

No one needs to label the desired scale and shift values by hand. During training, the model sees ordinary examples such as an image, a question, and the correct answer. It makes a prediction, measures the error, and sends that error back through both paths.

Over many examples, the main path learns useful visual features, while the conditioning path learns how questions should adjust those features. A question about color may learn one pattern of controls; a question about quantity may learn another. The only supervision is the final task itself.

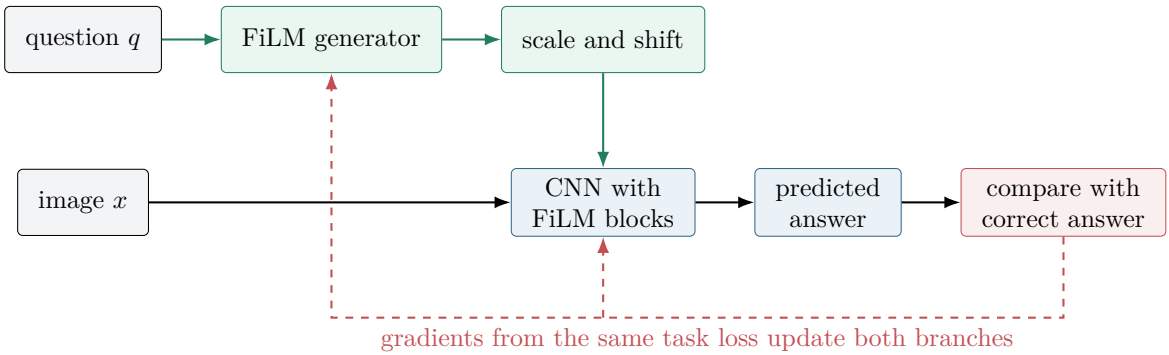

Figure 3: FiLM is trained end to end. The final error teaches both the content network and the network that produces the controls.

5 FiLM is related to normalization, but not tied to it

FiLM is often introduced next to BatchNorm or LayerNorm, which can make the idea sound more complicated than it is. Normalization and FiLM answer different questions:

- **Normalization** puts features on a more regular numerical scale.

- **FiLM** changes features according to the current context.

BatchNorm and LayerNorm often include a learned affine scale and shift after normalization. Conditional normalization makes those affine parameters depend on context. FiLM generalizes the same idea: the context-dependent transformation may be placed after normalization, as in the block above, but it can also be used elsewhere without normalization.

The same idea extends beyond CNNs. In a Transformer, the modulated features may be hidden dimensions rather than image channels. A global conditioning vector—such as a task descriptor, class label, or diffusion timestep—can generate scale and shift values for those dimensions.

6 When FiLM is a good fit

FiLM is appealing when the context is global and should influence many parts of the main computation. Examples include:

- answering different questions about the same image;
- adapting a model to different tasks or domains;
- controlling style, speaker identity, or generation time steps; and
- injecting an instruction into several layers of a shared backbone.

Its main strength is efficiency. The conditioning network only needs to produce a small number of controls, so the expensive backbone can be reused.

Its simplicity is also a limitation. Standard FiLM controls whole feature channels. It does not directly connect a particular word to a particular image location. If fine word-to-region alignment is the central problem, cross-attention may be more appropriate. In many systems the two ideas are complementary: attention decides *where* to look, while FiLM changes *how* the selected features are processed.

7 The big picture

FiLM can be understood without thinking about a particular architecture or a long derivation. One input provides the content. Another input provides the context. The context predicts a scale and a shift, and those two controls reshape the content features inside the network.

That small operation is the entire idea:

$$\textbf{new feature} = \textbf{context-dependent scale} \times \textbf{old feature} + \textbf{context-dependent shift.}$$

The result is a shared network that can adapt its internal computation to each question, instruction, style, task, or other source of context.

References

[1] Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. *FiLM: Visual Reasoning with a General Conditioning Layer*. AAAI Conference on Artificial Intelligence, 2018. https://arxiv.org/abs/1709.07871.

[2] Vincent Dumoulin, Ethan Perez, Nathan Schucher, Florian Strub, Harm de Vries, Aaron Courville, and Yoshua Bengio. *Feature-wise Transformations*. Distill, 2018. doi:10.23915/distill.00011.