1. **After each stride-2 conv, why do we double the number of filters?**

**A stride 2 conv with the default padding (1) and ks (3) will reduce the activation map dimension by half. Formula: (n + 2\*pad - ks)//stride + 1. As the activation map dimension reduces by half we double the number of filters. This results in no overall change in computation as the network gets deeper and deeper.**

1. **Why do we use a larger kernel with MNIST (with simple cnn) in the first conv?**

In Convolutional Neural Networks (CNNs), the practice of starting with a small number of kernels in the early layers and gradually increasing the number of kernels in deeper layers is a common design choice. This approach is rooted in several reasons:

1. **Feature Hierarchy**: CNNs are designed to learn hierarchical representations of features in the input data. The early layers of a CNN capture low-level features such as edges, textures, and simple patterns. By using a small number of kernels in the initial layers, the network can focus on learning these basic features that are present in the input data.
2. **Computational Efficiency**: Using a small number of kernels in the early layers helps control the computational cost of the network, as the number of parameters and computations required is lower compared to having a large number of kernels in all layers. This can be particularly important when working with large datasets and complex models.
3. **Generalization**: Starting with a small number of kernels can help prevent overfitting, as it imposes a form of regularization by limiting the complexity of the learned features in the early layers. This can lead to better generalization performance on unseen data.
4. **Information Abstraction**: As the network goes deeper, the number of kernels is increased to allow the model to learn more abstract and complex features that build upon the low-level features learned in the early layers. By gradually increasing the number of kernels, the network can capture increasingly sophisticated patterns and structures in the data.
5. **Dimensionality Reduction**: Increasing the number of kernels in deeper layers can help reduce the spatial dimensions of the feature maps while increasing the depth (number of channels). This can help in creating a more compact and informative representation of the input data as it progresses through the network.

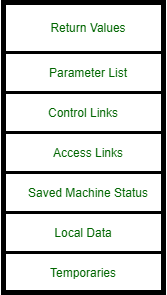
Overall, the strategy of starting with a small number of kernels and increasing them in deeper layers helps CNNs effectively learn hierarchical representations of features in a computationally efficient manner, leading to improved generalization performance and feature abstraction.

**3. What data is saved by ActivationStats for each layer?**

[**Activation stats**](https://www.geeksforgeeks.org/runtime-environments-in-compiler-design/):

**An activation record is a contiguous block of storage that manages information required by a single execution of a procedure. When you enter a procedure, you allocate an activation record, and when you exit that procedure, you de-allocate it. Basically, it stores the status of the current activation function. So, whenever a function call occurs, then a new activation record is created and it will be pushed onto the top of the stack. It will remain in stack till the execution of that function. So, once the procedure is completed and it is returned to the calling function, this activation function will be popped out of the stack.  
If a procedure is called, an activation record is pushed into the stack, and it is popped when the control returns to the calling function.**

**Activation Record includes some fields which are –   
Return values, parameter list, control links, access links, saved machine status, local data, and temporaries.**

****

***Activation Record***

**Temporaries:**

**The temporary values, such as those arising in the evaluation of expressions, are stored in the field for temporaries.**

**Local data:**

**The field for local data holds data that is local to an execution of a procedure.**

**Saved Machine States:**

**The field for Saved Machine Status holds information about the state of the machine just before the procedure is called. This information includes the value of the program counter and machine registers that have to be restored when control returns from the procedure**.

**4. How do we get a learner's callback after they've completed training?**

**Callback.\_\_call\_\_**

Callback.\_\_call\_\_ (event\_name)

*Call self.{event\_name} if it’s defined*

One way to define callbacks is through subclassing:

class \_T(Callback):

def call\_me(self): return "maybe"

test\_eq(\_T()("call\_me"), "maybe")

Another way is by passing the callback function to the constructor:

def cb(self): return "maybe"

\_t = Callback(before\_fit=cb)

test\_eq(\_t(event.before\_fit), "maybe")

[Callback](https://docs.fast.ai/callback.core.html#callback)s provide a shortcut to avoid having to write self.learn.bla for any bla attribute we seek; instead, just write self.bla. This only works for getting attributes, *not* for setting them.

mk\_class('TstLearner', 'a')

class TstCallback(Callback):

def batch\_begin(self): print(self.a)

learn,cb = TstLearner(1),TstCallback()

cb.learn = learn

test\_stdout(lambda: cb('batch\_begin'), "1")

If you want to change the value of an attribute, you have to use self.learn.bla, no self.bla. In the example below, self.a += 1 creates an a attribute of 2 in the callback instead of setting the a of the learner to 2. It also issues a warning that something is probably wrong:

learn.a

1

class TstCallback(Callback):

def batch\_begin(self): self.a += 1

learn,cb = TstLearner(1),TstCallback()

cb.learn = learn

cb('batch\_begin')

test\_eq(cb.a, 2)

test\_eq(cb.learn.a, 1)

/tmp/ipykernel\_5201/1369389649.py:29: UserWarning: You are shadowing an attribute (a) that exists in the learner. Use `self.learn.a` to avoid this

warn(f"You are shadowing an attribute ({name}) that exists in the learner. Use `self.learn.{name}` to avoid this")

A proper version needs to write self.learn.a = self.a + 1:

class TstCallback(Callback):

def batch\_begin(self): self.learn.a = self.a + 1

learn,cb = TstLearner(1),TstCallback()

cb.learn = learn

cb('batch\_begin')

test\_eq(cb.learn.a, 2)

**Callback.name**

Callback.name ()

*Name of the*[*Callback*](https://docs.fast.ai/callback.core.html#callback)*, camel-cased and with ‘Callback’ removed*

test\_eq(TstCallback().name, 'tst')

class ComplicatedNameCallback(Callback): pass

test\_eq(ComplicatedNameCallback().name, 'complicated\_name')

**TrainEvalCallback**

TrainEvalCallback (after\_create=None, before\_fit=None, before\_epoch=None,

before\_train=None, before\_batch=None, after\_pred=None,

after\_loss=None, before\_backward=None,

after\_cancel\_backward=None, after\_backward=None,

before\_step=None, after\_cancel\_step=None,

after\_step=None, after\_cancel\_batch=None,

after\_batch=None, after\_cancel\_train=None,

after\_train=None, before\_validate=None,

after\_cancel\_validate=None, after\_validate=None,

after\_cancel\_epoch=None, after\_epoch=None,

after\_cancel\_fit=None, after\_fit=None)

**5. What are the drawbacks of activations above zero?**

**Sigmoid**:-

**Normally used as the output of a binary probabilistic function.**

**Advantages:**

**-> Gives you a smooth gradient while converging.**

**-> One of the best Normalised functions.**

**-> Gives a clear prediction(classification) with 1 & 0.**

**Tanh**:-

**Normally used as the input of a binary probabilistic function.**

**Advantages:**

**-> Zero-centric function unlike Sigmoid.**

**-> It is a smooth gradient converging function.**

**RELU**:-(Rectified Linear Unit)

**Advantages:**

**-> Can deal with Vanishing Gradient problem.**

**-> Computationally inexpensive function(linear in nature).**

**ELU**:- (Exponential Linear Unit)

**Advantages:**

**-> Gives smoother convergence for any negative axis value.**

**-> For any positive output, it behaves like a step function and gives a constant output.**

**SWISH**:-

**Also known as self gated function. This activation function is one of the kinds that is being inspired by the use of the Sigmoid function inside an LSTM(Long Short Term Memory) based network.**

**Advantages:**

**-> Can deal with Vanishing Gradient problem.**

**-> The output is a workaround between RELU and Sigmoid function which helps in normalising the output.**

**SoftPlus**:-

**Advantages:**

**-> Convergence of gradient is smoother than RELU function.**

**-> It can handle the Vanishing Gradient problem.**

**6.Draw up the benefits and drawbacks of practicing in larger batches?**

Modifying the batch size during training as the performance improves can have both advantages and disadvantages. Here are some of the key points to consider:

Advantages:  
1. **Speed and Efficiency**: Increasing the batch size can lead to faster training times. Larger batch sizes allow for more examples to be processed in parallel, potentially speeding up the training process.

1. **Generalization**: Increasing the batch size can sometimes lead to better generalization. Larger batch sizes can provide a more stable estimate of the gradient, which can help the model generalize better to unseen data.
2. **Resource Utilization**: Increasing the batch size can lead to better utilization of computational resources. With larger batch sizes, the hardware resources like GPU memory can be used more efficiently.
3. **Convergence**: Adjusting the batch size can help the model converge faster to a good solution. By tuning the batch size, you can find a balance between convergence speed and generalization performance.

Disadvantages:  
1. **Overfitting**: Increasing the batch size can sometimes lead to overfitting, especially if the model starts memorizing the training data instead of learning general patterns. This can happen if the model is too large relative to the batch size.

1. **Memory Constraints**: Larger batch sizes require more memory, which can be a limitation, especially when working with limited computational resources like GPU memory.
2. **Learning Rate Sensitivity**: Changing the batch size can affect the learning rate that is optimal for training. The learning rate that works well for a small batch size may not be suitable for a larger batch size, and finding the right learning rate can be challenging.
3. **Stability**: Larger batch sizes can sometimes lead to instability during training, especially if the model is sensitive to the batch size. This can manifest as oscillations in the loss function or difficulties in convergence.

In practice, it's often recommended to tune the batch size along with other hyperparameters like learning rate, network architecture, and regularization techniques to find the best combination for your specific problem. Experimentation and monitoring the training process are key to understanding how modifying the batch size impacts the training performance.

**7. Why should we avoid starting training with a high learning rate?**

The learning rate controls how quickly the model is adapted to the problem. Smaller learning rates require more [training epochs](https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/) given the smaller changes made to the weights each update, whereas larger learning rates result in rapid changes and require fewer training epochs.

A learning rate that is too large can cause the model to converge too quickly to a suboptimal solution, whereas a learning rate that is too small can cause the process to get stuck.

The challenge of training deep learning neural networks involves carefully selecting the learning rate. It may be the most important hyperparameter for the model.

**8. What are the pros of studying with a high rate of learning?**

1. **Higher Earning Potential**

Earning a college degree opens the door to better income potential. If you are employed in a field that does not require a degree, having one will make it more likely that you will advance in the company you work for. On average, people with a college degree make more than those with just a high school diploma. A 2021 survey published by [Forbes](https://www.forbes.com/sites/michaeltnietzel/2021/10/11/new-study-college-degree-carries-big-earnings-premium-but-other-factors-matter-too/?sh=53cd7ae135cd) found that adults with a bachelor’s degree earned over twice the amount as those without one over their lifetimes.

1. **Better Employability**

Having a college degree increases the likelihood you will also have a job. According to the [Department of Education’s National Center for Education Statistics](https://nces.ed.gov/fastfacts/display.asp?id=561), the employment rate for 25 to 34-year-olds drastically increases with higher levels of education. The employment rate was 86 percent for those with a bachelor’s degree or higher as of March 2020, compared to 69 percent for those who had a high school diploma only.

1. **Preparation for a Specialized Career**

You can get a job without a college degree, but you may not be able to launch a career. Many specialized career paths require a degree as the foundation for entry (and success) in that field. If your chosen field has specialized job training requirements, earning a degree is the logical first step. For example, if you want to work in education or the medical fields, you will need a degree to get started.

1. **Increased Productivity**

When you earn a college degree, you develop more skills in your chosen field. More skills mean a greater level of productivity, which benefits everyone in the workforce, from the employee to the employer. This involves gaining skills in your field of study that will help you manage your time on the job. Furthermore, a degree usually comes with some hands-on training, increasing your aptitude.

1. **Better Communication Skills**

Higher education brings with it skills to communicate both verbally and in writing. You will be able to interact with your coworkers clearly, and learn how to communicate with your management team effectively. If your job involves speaking to customers or students, you will also gain important skills to help you do so. Clear written and verbal communication is critical to finding success at work.

1. **Improved Self Confidence**

Finishing college is a huge accomplishment. When you graduate, you gain confidence knowing you have done hard work and it has paid off. This  is vital in the workplace. Employers notice it in job interviews and when considering people for leadership roles. It is invaluable as you seek career advancement.

1. **Development of Critical Thinking Skills**

High school classes are usually focused on imparting knowledge and building basic skills, leaving little room for developing critical thinking. Yet in the workforce and in life, critical thinking is vital. By the time someone heads to college, they are usually ready to think on a deeper  level about various topics.

Many college programs have a heavy emphasis on critical thinking, helping you fine-tune your abilities to think clearly about the challenges you will face in your job.

1. **Creating Networking Opportunities**

When it comes to landing a job, it’s often all about who you know. A professional network is a place where new job opportunities are found. Going to college automatically expands your professional network, as you can build relationships with students, teachers, and others in your field.

Using tools like LinkedIn and other social media platforms, you can stay connected with those people after graduation. Over time, they can lead to information about potential positions or other networking opportunities.

1. **Creates Pathways to Career Advancement Opportunities**

Your college education not only grows your network, but it also starts you on the path toward career advancement opportunities. You may learn about conferences and events in your field that  add to your knowledge base and your professional connections. You may also be able to receive a certification that opens the door to greater career advancement.

In addition, having a college degree gets your foot in the door for many promotions that simply aren’t available without one. If you want to move forward in your career, this is the key steppingstone.

1. **A Happier Life**

Greater income potential, greater career advancement, a strong network and higher self-confidence all add up to helping you create a happier life. Research backs up this claim, too. In a [2016 Pew Research Study](https://www.pewresearch.org/social-trends/2016/10/06/3-how-americans-view-their-jobs/), 23 percent of adults with limited education said they were not happy with their lives compared to just nine percent of those with a bachelor’s degree or higher.

1. **Why do we want to end the training with a low learning rate?**

**It is typical to start training with a higher learning rate and then decrease it either after a certain number of iterations or gradually at every step. This allows the network to first take faster steps toward the optimal solution, and when converging, focus on finer details without oscillations.**