**Sanyam:**From our earlier conversation, we both agreed on the viewpoint of Online learning being an equally good if not better alternative to a Masters. Can you share your thoughts on this?

Do you think Kaggle can prepare you for a Data Science position better than a Masters?

**Aakash:**There are two types of online courses available on the Internet today. The first ones are those that cover the “width” while the second one are those that cover the “depth”. For example, Udacity courses cover the width. They will teach you almost all the aspects but none of them will be covered in depth. On the other hand, coursera courses are great if you are looking to deep dive into some aspects but they won’t cover every element required. This the major bottleneck today. Finding all the things that are required to learn in a single place is a bit difficult when it comes to an online course. Plus nothing is cheap. Online courses are not that expensive as Masters but they are expensive.

Competing on Kaggle and opting for Masters are two totally different things. The former one makes you better at trying different things and being recognized to a wider audience while the latter one makes you good, if not better, at research. Some people love to be in college again while other prefer the hands-on experience and solving problems on a daily basis.

**Sanyam Bhutani:**You’ve had an interesting career path. After completing your PhD you started your first company and later decided to switch to teaching the tech that you had worked on over the years.

What made you start the PyImageSearch blog. Why was it important to you?

**Adrian Rosebrock:**There are actually two reasons I started PyImageSearch.

The first is I had built and launched a two computer vision and machine learning startups while in college. It was a fun experience and I learned a lot during the process. But I always missed writing as well.

The second is that I was self-taught much of my computer vision education. At my school there were some incredible professors doing work in machine learning and graphics, but no one in particular was dedicating their work to the computer vision field.

I took my first (and only) computer vision-related course during my final year of my undergraduate career from a part-time instructor. I then took three separate machine learning and data science courses.

Everything else, including the practical elements of how to “glue” all the pieces together and build an actual computer vision solution with OpenCV, was something I had to learn on my own.

It was a lot of hard work and back then the documentation was hard to find, incorrect, or “only okay” at best. Furthermore, practical, hands-on tutorials were few and far between.

During the final semester of my graduate school career I decided to start PyImageSearch. I felt I had a lot to share and I really wanted to help others who were either (1) self-taught as I was or (2) trying to find practical tutorials to supplement their university courses/education.

I also knew that I wanted to write a book regarding computer vision but I wasn’t sure what the topic would be. Starting the blog helped me learn from my readers and ultimately lead to my first book, [*Practical Python and OpenCV.*](https://www.pyimagesearch.com/practical-python-opencv/)

I write a lot of technical content on the blog and in my books/courses so it was fun to create something light-hearted and fun while at the same time actually teaching readers how they could utilize OpenCV, Keras, and Deep Learning in their own projects.

**Sanyam Bhutani:**I have to say, like many others, I’m a fan of your articles.

Could you share some tips on effectively writing technical blog posts?

**Adrian Rosebrock:**One of the worst ways to start writing anything, whether technical or not, is to open a new document and expect that you’ll have words pouring out of you, magically filling up the page — it rarely works like that and it’s normally a recipe for frustration and failure.

Instead, I recommend outlining first. I personally outline in bullet points.

Start with your headers first — what are you going to cover? And in what order?

From there you back and start filling in the sub-headers. What do you need to cover in each section?

Finally, I go back and include the details for each section. Sometimes I even write in complete sentences, but that’s a personal choice.

I always leave notes to myself such as “Insert an image here” or “Lines X-Y of some\_script.py go here”. I try to keep myself focused on the actual writing process as much as I can. The actual images and code can be inserted later during typesetting.

**Sanyam Bhutani:**You’ve always used a “theory minimum” approach when teaching a concept. Why do you think a code-first approach is useful when both-learning and building a project?

**Adrian Rosebrock:**I think “some” theory can go a long way when implementing a computer vision or deep learning algorithm but when you go to colleges or universities that’s all you find in the textbooks. Lots of mathematics. Lots of equations. That is only one side of the coin though — those books are missing the practical implementation.

The best way to learn these concepts is to intersect the two. Teach a bit of theory then show how it’s done in code. Many books and textbooks purposely try to separate the two — in my opinion that is the incorrect approach.

The fact we live in a world where developers and engineers can now type `import sklearn` or import keras` has changed the machine learning landscape.

The notion that people need a decade of mathematics or a college degree in computer science to perform machine learning is frankly incorrect.

The best way to get started in machine learning is to get started. Install one of the Python machine learning or deep learning packages. Follow the basic tutorials and examples the best you can. Train your first machine learning model and look at the result.

Did it get you excited? Do you want to learn more? Awesome, that’s great! Go dig into the documentation or other tutorials where you’ll continue to learn.

If you didn’t like it, that’s fine too. It’s better that you learn now that you don’t enjoy coding or writing machine learning code before you spend a year studying it.

Don’t get me wrong — theory is very important, especially if you wish to write scientific publications. But if you’re a programmer just getting started with machine learning or deep learning, start with existing code-based tutorials first. See if you like it and enjoy it — then supplement your education with theory as you go along.

How do you manage your time so effectively? Why are these contributions important to you?

**Andrew Trask:**Ha! I’m still working on that to be honest. The best advice I can give there is to try to find activities where you get synergies of scope. For me that’s a focus on the intersection of cryptography and deep learning. The book, OpenMined, and my research all focus on it, so when I’m working on one I’m also (more or less) working on the rest as well. It doesn’t always work out, but its easier than if all three were totally separate.

**Sanyam Bhutani:**You’re an advocate of Decentralised AI.

Why do you feel it’s important at such an early stage — even when the AI systems aren’t very production ready yet?

**Andrew Trask:** AI is like any other industry. There’s a supply chain. For AI, it starts with people who create data (normal people like you and me), which is then collected by various applications, aggregated through data markets, refined into AI models, and then sold in the form of either direct insights or models themselves. At the moment, everyday people only really control the very first piece of that value chain. The decentralization/privacy movement is about creating technology which allows everyday people to have ownership/control over a larger portion of that chain. Over time, this increases the chances that the entire system creates value for everyday people, that it works in their interests. Furthermore, the longer one waits to re-direct ownership of a value chain, the more difficult it is to move. To that end, the sooner the better.

**Sanyam Bhutani:**Apart from Decentralised AI, what do you feel is another aspect that the practitioners must pay attention to?

**Andrew Trask:** Security and performance is the most frequently overlooked limitation — also a realistic end user story. Many people make the mistake of starting with a “sexy idea” and then working backward to the customer. We need to start with customer needs and then figure out how decentralization can meet those needs better than centralization.

I began my journey into Deep Learning 6 years ago. It’s not that long. Deep Learning isn’t like Physics where it takes a whole career to get to the bleeding edge. It’s a relatively new field and it moves quite quickly which means that it’s a bit biased toward new folks coming into the field who have the time to keep up with what’s going on. Also, once you **really**understand backpropagation, stochastic gradient descent, convolutions, attention, and LSTMs you’ve got most of the big parts down (you know enough to work professionally in deep learning). Many other advances are merely modifications, permutations, or extensions of these concepts.

**Sanyam Bhutani:**As a Domain Expert, what are your thoughts of using an ML Algorithm in an application where a Human Life is under consideration?

How can we ensure that we’re building the right tools or what steps should we take to ensure the Technology works in a positive manner?

**Dr. Alexandre Cadrin-Chenevert:**For the available resources, most societies want the best health outcome for their population.

When accessibility is not a huge problem, you basically want to improve the quality of health services to improve the outcome. There is an incredible potential to improve the quality of health services by combining the work of a medical expert with the output of an ML algorithm. For example, in radiology just lowering the variability of human interpretation can be a significant beneficial step.

When accessibility is the main bottleneck, then ML algorithms can potentially help to triage the population into different levels of health priority. This ultimately can improve accessibility and outcome of your population with the same amount of human and financial resources.

Of course, there is a significant statistical trap with machine learning algorithms. And this trap is even wider with deep learning algorithms. Data scientists usually present performance metrics of a trained algorithm based on an unseen test dataset. This is the usual proof of performance generalizability of an algorithm. But in healthcare, in clinical practice, there are tons of non-controlled and hidden variables in the data that can be very different compared to the test data distribution. These variables are related to genetic, gender, age, risk factors, symptoms, types of acquisition machines and many more. This is inducing a bias between the test and the clinical data. And this bias is frequently not directly measurable. But this bias in clinical data can generate very large variations of performance of the ML algorithm.

So, I really recommend to anyone who considers using an ML/DL algorithm in clinical practice to strongly validate the reported performance on its own data. But this validation process can be very time consuming and difficult because most of the clinical institutions are not organized to do this kind of validation. But, I am definitely convinced about this need; I would even recommend explicit local validation even if an algorithm is already cleared by national regulators.

Participating in Kaggle competitions was of paramount importance in my learning curve of deep learning. Maturation of almost any learning process is based on the transformation of knowledge to skills, or concept to actions. Each competition is an opportunity to evolve in this direction. But, naturally, the tradeoff is the time you invest in these competitions. So my own perspective was to participate in a limited number of computer vision competitions selected to catch efficiently most of the potential benefit.

It is an ideal scenario to apply the needed iterative process of trying experiments by yourself to solve the problem, and then coming back to the forums to learn from the kernels and the forums posts. If you discover an interesting kernel then try to deeply understand what it is doing and why it was written that way. Some of these kernels are hidden gems and you need to meticulously unpack them to discover the value.

**Sanyam Bhutani:**Before we conclude, any advice for the beginners who even though are excited about the field, feel overwhelmed to even get started with Deep Learning?

**Dr. Alexandre Cadrin-Chenevert:**To conclude, I’ll cite this guy, named Steve Jobs, who is a lot better than myself to give advices:

“You’ve got to find what you love. And that is as true for your work as it is for your lovers. Your work is going to fill a large part of your life, and the only way to be truly satisfied is to do what you believe is great work. And the only way to do great work is to love what you do. If you haven’t found it yet, keep looking. Don’t settle. As with all matters of the heart, you’ll know when you find it. And, like any great relationship, it just gets better and better as the years roll on. So keep looking until you find it. Don’t settle.”

**Sanyam Bhutani:**For the readers and noobs like me who want to become better kagglers, what would be your best advice?

**Artur Kuzin:**Perhaps I have a somewhat alternative point of view. But strongly believe that the most important thing is the ability to desire and to be obsessed with something. This is the ability to get interested, not to surrender halfway, put all on till the end and fight to the last second. If you have the desire, then you will understand how to become the best.

**Sanyam Bhutani:**H20.ai is working on many exciting projects, could you tell us more about your role at H2o.ai?

**Dr. Bojan Tunguz:**I work with the engineering team where I help with the development of DriverlessAI, as well as with our marketing, sales, and other outward-facing teams in their effort to promote our products, services, and the general ML approach and philosophy. I’ve been particularly excited about our recent educational initiative since it dovetails well with my former background in academia. I am also pretty involved with our efforts in the underwriting industry, where I bring my previous professional experience. H2O is a great organization for me to work at since it allows for the full spectrum of my talents and interests to be valued and utilized.

**Sanyam Bhutani:**It’s quite interesting to see that as an artist you’re trying to create Neural nets that replicate human creativity.

From an Artist’s perspective, What are your thoughts about making a creative “AI” ?

**Christine McLeavey:**That’s an interesting question. On one hand, it’s hard to see an LSTM as creative. Behind the scenes, my model is just predicting how likely it thinks each note is for the upcoming step. It then randomly chooses between those notes, based on that probability. I can increase the randomness and maybe it will seem more creative, but it’s still just based on patterns in the training data.

On the other hand, maybe human creativity is similar in ways. We all have building blocks that we put together into patterns — often usual, predictable ones, and then sometimes wacky random ones. Maybe the most important thing about human creativity is our ability to recognize the times when our wacky and random experiments are actually beautiful art.

I’m excited to see how “creative AI” will start interacting with human emotions. Can a music generator make a piece that makes me feel sad? Can a neural net decode the news, and create a painting that captures our reaction to current events?

In terms of specifics — one of my favorite pieces of advice was from FastAI’s Jeremy Howard. He told me to focus on making one really great project that showed off my skills. He says that so many people have many interesting but half-finished projects on github, and that it’s much better to have a single one that is really polished.

**Sanyam Bhutani:**Before we conclude, any advice for the beginners who even though are excited about the field, feel overwhelmed to even get started with Deep Learning?

**Christine McLeavey:**I would say: embrace that feeling of being overwhelmed. So often, when I first tackle a new subject, I have the feeling that all the ideas are jumbled in my head and that I’m not really understanding anything. Nowadays, I’ve gone through this so many times, that I recognize that feeling of being overwhelmed, and I’ve learned not to give up at that point. I know that when I come back to the material in a day or a week, it will make more sense.

I’m always looking for the next problems to solve and the next product to work on, it’s a lot of fun.

**Sanyam:**You’ve mentioned some of the great platforms that you’re working on, and I personally know that you ship code very fast.

How do you manage to work on multiple task? What’s your secret?

**Dominic:**When I wake up, I always know what I’ll work on for the day. I plan out my whole day the evening before, using Todoist. I work full-time, which takes up most of my day, so between meals, going for walks, working out and sleeping, I need to know what I can get done for the day. The key is to move fast and not being afraid of shipping things.

I usually have a large list of to-dos for each of my projects. Some are more important and usually get done quickly, some others have been on that list for months. It’s all about managing expectations, splitting up work in chunks and shipping fast.

This is something that people have lost sight of nowadays, but end-to-end differentiable models trained with backpropagation are just one solution to the problem of learning modular-hierarchical representations for perception, and there are alternative avenues that haven’t been explored very much. And that problem itself is just one of the many subproblems that compose the field of AI.

**Sanyam Bhutani:**What are your thoughts about Keras being made the default API in TensorFlow 2.0. Why do you feel it is necessary?

**François Chollet:**TensorFlow is an extremely powerful framework, but it has long suffered from usability issues, in particular a sprawling and sometimes confusing API. TensorFlow 2 fixes these issues in a big way. Two things are at the center of this effort: eager execution, and the Keras API. Eager execution brings an imperative coding style to TensorFlow, making it more intuitive and easier to debug. The Keras API consolidates usage patterns into one coherent spectrum of really productive and enjoyable workflows, suited to a variety of user profiles, from research to applications development to deployment. I’m really excited about what we’re about to release. But you’ll see for yourself soon!

**Sanyam Bhutani:**For the readers and the beginners who are interested in working on Deep Learning with the dream of working at Google someday, what would be your best advice?

**François Chollet:** I don’t think you should tie your dreams to external markers of status, like working for a specific big-name company, or making a specific amount of money, or attaining a specific job title. Figure out what you value in life, and then stay true to your values. You’ll never have to regret a single decision.

**Sanyam Bhutani:**Do you feel Machine Learning has been overhyped?

**François Chollet:**Definitely, to some extent. I think that machine learning is, in a way, simultaneously overhyped and underrated. On one hand, people tend to vastly overestimate the intelligence and the generalization power of current machine learning systems, perceiving machine learning as a kind of magic wand that you can wave at arbitrary problems to make them disappear. This is, of course, largely false, there is very little actual intelligence in our algorithms, and their scope of application is extremely narrow. But at the same time, most people still underestimate how much can be achieved with the relatively crude systems we have today, if we apply them systematically to every problem they can potentially solve. Machine learning is, in a way, the steam power of our era: a pretty basic mechanism that nonetheless has the potential to profoundly change the world when used at scale.

The flood of content being published today may look important, but most of it is noise. Focus on the big questions.

**Sanyam Bhutani:**I think you’re the Thomas Edison of GAN(s) (or at least photo colorization using DL)-the idea didn’t work for quite a few weeks and you had quite a bit of unsuccessful experiments (over a 1000).

What made you stick to the project and not give up? How do you think can a software engineer stay motivated and not give in to the “imposter syndrome”?

**Jason Antic:**Well that comparison to Edison is rather flattering! So, what made me stick to the project and not give up is this somewhat unreasonably optimistic view of mine that there’s a solution to any reasonable problem and that it’s just a matter of effectively navigating the search space to find the answer. Effectively to me, that means doing a lot of experiments and being methodical, and to constantly question your assumptions and memory because that’s typically where problem-solving goes wrong.

That being said, despite my undeniable successes I still to this day fall into that dark mental state of self-doubt, wanting to give up, and “imposter syndrome”. Even earlier this week it started creeping up on me again when I was running into difficulties, and the intrusive thoughts started pouring in again. “You’re deluded, and you were just lucky with DeOldify.” Believe it or not that still happens.

Then I pushed through it and figured it out, and I am very excited about what will be released in the next month as of this writing :)

How do I push through it? The belief that a solution is there and that I’m capable of finding it simply because I’m a normally functioning human being that can problem solve is a big one. That’s the key point here — it’s not so much a matter of intelligence as it is of the method (and that’s learnable!). Another motivating factor is the realization that there is in my mind no better way to spend my time then to try to solve big/cool problems, and that it’s worth the blood, sweat, and tears. Purpose and autonomy are huge motivators.

**Sanyam Bhutani:**There is a flipside to it as well, how does someone know when to quit a project that might just be too ambitious for the given technology?

**Jason Antic:**Yes, you definitely have to know when to quit, and that’s quite the art. I say “No” to, and/or quit, a lot of things actually. Why? Because for everything you say “Yes” to, you’re saying “No” to many other things. Time (and sanity) is precious. As a result, especially lately, I’ve said “No” to quite a few opportunities that others find crazy to say “No” to.

So quitting for me is decided first on whether or not the path falls squarely in my realm of values, interests, and maintaining sanity. If you’re talking about an ambitious technological project, then you have to go one step further and evaluate whether or not it’s actually feasible. Often you simply can’t answer that right away, especially if you’re doing an ambitious project! So that’s why you experiment. If you discover a sound reason (and not just a rage-quit rationalization) as to why something won’t work, then quit, learn from it, and move on! But be careful on this point — a lot of problems are solved handily simply by shifting perspective a bit. For this reason, I’ll stick with a problem a bit longer than seems reasonable, because often my perspective is simply wrong. Often the solution comes when I walk away (shower thoughts, basically).

Hence my advice is this: Find something you’re interested in enough to pursue it in a manic way and guide your efforts by at least somewhat rigorous experimentation. And stay the course until you have the actual reason (evidence) to believe that what you’re pursuing is impossible as opposed to just unknown. I think this is where most people shoot themselves in the foot — they give up way too easily!

**Sanyam Bhutani:**For the readers that are curious about what does a day in the life of a researcher look like, can you give us an insight?

How much time do you spend on Experimenting Vs Exploring new ideas?

**Dr. Leslie Smith:** I don’t know about other researchers but a majority of my time is spent on reading, experimenting, writing, email and talking with people. Reading and experimenting are the catalysts for most of my ideas.

**Sanyam Bhutani:**​I want to go back a little bit and ask about your background.

With almost no coding background, you quickly picked up Machine Learning and became well-established in the field.

Could you tell us more about your journey?

The common belief is that one needs dozens of years of coding experience to even get started with ML.

**Mamy André-Ratsimbazafy:**I basically learned to code beyond “exercises” while doing ML. Like many, I often had to look into project documentation (especially Pandas) and Stack Overflow but in any case, what helped me is just knowing how variables work, what is a for loop, and if-then branch and a function. What we use in ML is in any case quite specific to numerical computing so what you learn on a web project, for example, won’t really help you beyond the basic programming construct and, very important, versioning your experiments. Now, being experimented will help you a lot when doing a Kaggle competition and your project starts getting huge to avoid you and your teammates getting lost in the code.

**Sanyam Bhutani:**​What are your thoughts about most of the Job postings requiring a Masters or Ph.D. level of expertise for ML?

Having a “non-traditional” background, how can one find work in this field?

**Mamy André-Ratsimbazafy:**I think most recruiters and companies are not mature enough to evaluate candidates. Many are still building up their teams and don’t have the in-house expertise to look for external signs of competence for recruiting. I’m much more concerned about the experience requirements. The rise of deep learning was in 2012 with AlexNet. Besides the US, I would guess that many data science focused masters were created around 2014 in universities so most new hires with an actual data science degree would have at most about 3 years of experience. Most more experienced people would probably be self-taught.

As I said at the start, companies are looking for signs of competence, the best way is to have a portfolio to show. Kaggle competitions and personal machine learning projects are an excellent way regarding that. Pick your favorite sport/movie genre/games/food, find a [dataset](https://www.kaggle.com/datasets) and analyze it. In the interview show how it relates to the problems of the company you are interviewing for. You will be in known land and would have worked on it for hours already.

Another important thing is networking. Before applying, try to talk to people working in the same role in several companies. For example, data scientist is a bit of a kitchen sink role, with every company having a different scope for them. How to talk to them? Just say that you’re quite interested in their day-to-day, their challenges, and responsibilities and offer them to meet over a coffee. LinkedIn is quite useful for that.

**Sanyam Bhutani:**You’re a Competitive Data Scientist and also an open source contributor to projects: StackNet, KazAnova.

Where does kaggle come in the picture? Is it related to your other projects and the research work?

**Dr. Marios Michailidis:**Kaggle helps me in various ways:

1. Learn new skills, new tools, what’s hot.
2. Solve a variety of problems.
3. Become part of a (very generous) and open community.
4. Collaborate with other experienced people in the field.
5. Test/prove my ideas. Benchmark myself and how well I can do in a variety of problems.
6. Receive recognition, promote my research.
7. Do my job better. The company I work for: [H2o.ai](https://www.h2o.ai/), is a leader in the development of software for data science and predictive analytics. Kaggle is a great environment to test our products and ensure they fair well against some of the top data scientists in the field.
8. Generally become better in my craft.

**Sanyam Bhutani:**​​ You’re currently working as a Data Scientist at Kaggle, you have a background in Linguistics. Could you tell us how did you get interested in NLP and Data Science?

**Dr. Rachael Tatman:**I definitely got into it from the “science” side. When I started grad school I had very little programming experience, just a couple intro CS courses in undergrad. My main research interests at the time were the effects of different elicitation tasks when collecting voice data (like reading text vs. holding a conversation) on the speech produced. Since these effects were pretty hard to tease out, I took some graduate level statistics courses to learn more about how to model them. This is where I was introduced to R. I kept using R for different research projects, learned a bit of MatLab for signal processing and played around a bit with Python because I was using Python software to run my experiments and had some pretty specific needs. With practice, I became more confident in my ability to write code to solve problems. Since my problems were generally around collecting, transforming and analyzing data, this is probably the point at which you could have started calling me a “data scientist”.

As for getting into NLP, as my research slowly changed over time I started working on problems that were more and more relevant to NLP. One of my projects, for example, looked at how people online use different spellings to show different dialects. Eventually, however, I realized that NLP researchers really don’t read linguistics papers; in order to join in the conversations going on, I started going to NLP conferences. Between the machine learning results that were being presented at conferences and the statistics courses I was still taking, I got up to the point where I could start reading and understanding machine learning papers within a year or two. By the time I graduated, I felt pretty comfortable calling myself an NLP researcher.

All in all, it was not a very efficient way to go about it. A lot of what I did in my degree was not at all relevant to what I do now. (I took several years of American Sign Language, for example, and wrote several research papers on sign language phonology.) To be fair, though, I had no idea I was going to be a data scientist when I went into grad school. In fact, the career didn’t even exist when I started my Ph.D.!

**Sanyam Bhutani:**​Many job boards (For DL/ML) require the applicants to be post-grads or have research experience.

For the readers who want to take up Machine Learning as a Career path, being a Ph.D. in your domain, do you feel having research experience is a necessity?  
What are your thoughts about kaggle as an experience factor?

**Dr. Rachael Tatman:**My universal advice is to *not*get a Ph.D. I even [wrote a blog post about it a while ago](https://makingnoiseandhearingthings.com/2016/04/28/should-you-go-to-grad-school-for-linguistics/). The blog’s about linguistics specifically, but most of it applies to machine learning as well. I think that having a Ph.D. can be an advantage when you’re looking for data science jobs, but unless you really want to 1) do research or 2) be a professor there’s no really no benefit to getting a Ph.D. that you can can’t get more quickly doing something else.

I think that Kaggle, or other practical experience, will get you to the point where you can apply for jobs much more quickly. I probably wouldn’t recommend *only*doing Kaggle competitions, though. You’ll learn a lot about algorithms that way, but you won’t get as much practice with things like cleaning data or designing metrics. That’s part of the reason I suggest that people work on their own projects as well. That shows off your ability to come up with interesting questions, source and annotate data, clean your data and think about what users want.

**Sanyam Bhutani:**I’m also a big fan of your live streams and kernels.   
Could you share a few tips on writing good kernels and becoming a better technical speaker?

**Dr. Rachael Tatman:**Hmm, what makes a kernel “good” is subjective, but the ones that really stick out for me are the ones that make me go “oh my gosh, I wish I’d thought of that!”. I really like to see people come up with new approaches for interesting problems, like [this kernel](https://www.kaggle.com/nateaff/finding-lego-color-themes-with-topic-models) that uses topic modeling, an NLP technique, to cluster LEGO sets based on their color.

As for technical speaking, the best two pieces of advice I can give you are, first, to practice as much as possible. Ask if you can give talks at local events or to relevant clubs. The more talks you give the less nerve-wracking they are and the more you learn what is effective for you. Practice is doubly important when you’re prepping a talk. I usually try to run through the talk at least twice a day in the week leading up to it, making little adjustments when I come across awkward places. Of course, I don’t do that with live streams. I pretty much treat livestreams like technical interviews; it doesn’t matter if I make mistakes so long as I’m telling you what I’m thinking so you can follow my thought processes.

My second piece of advice is to be as specific as possible. One of my personal pet peeves are talks that are about how “data science is revolutionizing something” but that is super vague. I want information I can actually apply! If you built a model that does X, talk about why X is important, how you built the model, what makes your model different from other models and how it performed in various situations. Tell me about what specifically you did that didn’t work so I know not to try it. Think about what you wanted to know about whatever you’re talking about a year ago and then tell me those things.

**Sanyam Bhutani:**​What developments in the field do you find to be the most exciting?

**Dr. Rachael Tatman:**Ooo, good question. I think the papers that I’m most excited about are the ones that offer theory-based explanations for why certain model architectures work better for certain problems. Empirical results, like “we tried x and it worked better than y”, are great, but I want to know more about *why*x and y are performing differently.

**Sanyam Bhutani:**​Before we conclude, any tips for the beginners who aspire to become Data Scientists and Kagglers but feel completely overwhelmed to even start competing?

**Dr. Rachael Tatman:**Celebrate failure! If you fail at things it’s because you’re pushing yourself and growing, and that’s a wonderful thing. If you try things and they don’t work, then you’re just getting closer to finding out what *will*work, whether that’s picking a better model architecture or just figuring out how to get this error message to stop showing up.

I also think we all, including me, compare ourselves to this shadowy “machine learning expert” who knows everything and also gets stuff right, but in reality, everyone only knows the tiniest little slice of everything there is to know. Don’t be afraid to ask questions and look things up if you don’t know them. (I search for things all the time while I’m coding!) But also don’t forget that you’ve got a lot of knowledge already. You’re bringing all your life experiences with you to learning and you never know what will end up leading to the next big breakthrough.

**Sanyam Bhutani:**I have to confess: As much as I’m a fan of the Top Down approach. Initially I found it difficult to follow fast.ai, I would spend too much time reading theory which would indeed be later taught by Jeremy in another lecture.

Most of us have been taught in a bottom up manner our entire student life, How can we adapt better to the “Top Down” approach?

**Dr. Rachel Thomas:**This is a good question! For those unfamiliar with the concept, math is traditionally taught in a “bottom up approach,” in which you have to learn each individual item you’ll be using before you can eventually combine them into something interesting, but many students lose motivation or drop out along the way. In contrast, areas like sports or music are often taught in a “top-down” way in which a child can enjoy playing baseball, even if they don’t know many of the formal rules. Children playing baseball have a general sense of the “[whole game](https://www.amazon.com/Making-Learning-Whole-Principles-Transform-ebook/dp/B0037NWZZ0)”, and learn the details later, over time. [We use this top-down approach at fast.ai](https://www.fast.ai/2016/10/08/teaching-philosophy/) to get people using deep learning to solve problems right away, and then we teach about the underlying details later as time goes on. Our approach was inspired by [Harvard professor David Perkins](https://www.amazon.com/Making-Learning-Whole-Principles-Transform-ebook/dp/B0037NWZZ0) and [mathematician Paul Lockhart](https://www.maa.org/external_archive/devlin/devlin_03_08.html).

I still find myself defaulting into a “bottom-up” approach sometimes, because it’s such a habit after 2 decades of traditional schooling. Using something when we do don’t understand the underlying details can feel uncomfortable, and I think the key is to just accept that discomfort and do it anyway.

**Sanyam Bhutani:**I also want to ask about your thoughts on AutoML: Do you think we’ll become obsolete and AutoML will eventually automate part of a data scientist’s toolbox or even the complete job?

**Dr. Rachel Thomas:**I think that we are already starting to automate parts of a data scientist’s toolbox, and that this can be a positive. Automated tools such as spell check and SwiftKey in other domains have been very useful!

As I wrote in my [series on AutoML](https://www.fast.ai/2018/07/16/auto-ml2/), I think that it is an incorrect focus to try to create products that completely automate data science (in part, because such attempts invariably [miss important components](https://www.fast.ai/2018/07/12/auto-ml-1/)), but that we should instead think of Augmented ML. Whereas AutoML is often focused on the goal of complete automation, the focus of *augmented ML* is on figuring out how a human and machine can best work together to take advantage of their different strengths. An example of *augmented ML* is Leslie Smith’s [*learning rate finder*](https://towardsdatascience.com/estimating-optimal-learning-rate-for-a-deep-neural-network-ce32f2556ce0) ([paper here](https://arxiv.org/abs/1506.01186)). The learning rate finder (a chart you look at to determine a good learning rate) is faster than AutoML approaches to the same problem, improves the data scientist’s understanding of the training process, and encourages more powerful multi-step approaches to training models.

I believe in all industries, tools are being created to allow workers to be more efficient. This can be good, when it entails automating work that humans find tedious or difficult. However, it is and will continue to have an impact on the number of jobs, since greater efficiency often allows for a smaller number of workers. I believe that societal and policy solutions (such as re-introducing competition, enforcing antitrust laws, addressing negative externalities, protecting human rights, a negative income tax, and universal basic income) are needed to address this.

**Sanyam Bhutani:**I’m a fan of your amazing blogposts.   
Could you share a few tips for the readers who want to become better (Tech) writers ?

**Dr. Rachel Thomas:**One [piece of advice](https://www.fast.ai/2017/12/18/personal-brand/) is to consider that your target audience is you-6-months-ago, not Geoffrey Hinton. What would have been helpful for your former self to hear? You are best positioned to help people one step behind you. Many experts have forgotten what it was like to be a beginner (or an intermediate) and have forgotten why the topic is hard to understand when you are first learning it. The context of your particular background, your particular style, and your knowledge level will give a different twist to what you’re writing about. I wrote a post on [getting started blogging](https://medium.com/@racheltho/why-you-yes-you-should-blog-7d2544ac1045).

I really appreciate when [Andrew Trask](https://hackernoon.com/interview-with-deep-learning-researcher-and-leader-of-openmined-andrew-trask-77cd33570a8c) spoke in his interview with you about the importance of high-quality blog posts and putting time into your writing. I am slightly embarrassed by how much time I put into many of my blog posts (I typically go through many iterations and re-writes), but it often pays off

I encourage everyone to learn math and technical topics on an “as-needed” basis. That is, start doing the work you are interested in doing, and if you come across some topic that you really need to be able to continue, learn it at that point. I don’t recommend trying to front-load all the math and technical topics that you think you may need, because in many cases you won’t need nearly as much as you think, and this can lead to students feeling bogged down or losing motivation.

Also, the fields of computer science and math are huge, so even someone with a “traditional, technical background” is only going to have studied some subset of the many, many computer science topics out there. For instance, my college education taught me how to prove if an algorithm was NP-complete or Turing computable, but nothing about testing, version control, web apps, or how the internet works.

Jeff Dean recently said during a talk at the Deep Learning Indaba that he thinks it’s better to read ten abstracts than one paper in-depth as you can always go back and read one of the papers in-depth. I agree with him. I think you want to read widely about as many ideas as possible, which you can catalogue and use for inspiration later. Having a good paper management system is key. I’ve been using Mendeley. Lately, I’ve been relying more on [Arxiv Sanity Preserver](http://www.arxiv-sanity.com/recommend" \t "_blank)to surface relevant papers.

**Sanyam Bhutani:**You also maintain a great blog, which I’m a great fan of.

Could you share some tips on effectively writing technical articles?

**Sebastian Ruder:**I’ve had the best experience writing a blog when I started out writing it for myself to understand a particular topic better. If you ever find yourself having to put in a lot of work to build intuition or do a lot of research to grasp a subject, consider writing a post about it so you can accelerate everyone else’s learning in the future. In research papers, there’s usually not enough space to properly contextualize a work, highlight motivations, and intuitions, etc. Blog posts are a great way to make technical content more accessible and approachable.

The great thing about a blog is that it doesn’t need to be perfect. You can use it to improve your communication skills as well as obtain feedback on your ideas and things you might have missed. In terms of writing, I think the most important thing I have learned is to be biased towards clarity. Try to be as unambiguous as possible. Remove sentences that don’t add much value. Remove vague adjectives. Write only about what the data shows and if you speculate, clearly say so.  
Get feedback on your draft from your friends and colleagues. Don’t try to make something 100% perfect, but get it to a point where you’re happy with it. Feeling anxiety when clicking that ‘Publish’ button is totally normal and doesn’t go away. Publishing something will always be worth it in the long-term.

**Sanyam Bhutani:**What suggestions do you have for beginners who want to write great kernels?

**Sudalai Rajkumar:**Kindly read multiple good kernels and try to understand them in detail. Learn how they create insights from the data, the plots they have used to portray the data, the inferences that they have come up with. It is also a good idea to take up a new concept (like a new algo or a novel technique) and educate people about the same. I personally do not like the kernels which just blends the output of two or three other kernels and get a high score.

**Sanyam:** What would you recommend to someone starting out in the field?

**Tuatini:** You have to show your skill to the world, one way or another, to justify your prices. If you charge a lot of money but have no presence on the internet (no blogs/no github/no review on your profile/no kaggle profile) etc…. It will be hard for you to justify those prices. Every person is different but for me what worked was writing a blog/doing some open source projects. Kaggle is good also to show your skills, I think it’s even better than blogging/open source as everyone can see and especially understand your skills based on your kaggle profile. I mean, take a non tech potential client and give him a github vs a kaggle profile, what stands out the most to him? A bunch of line of code/projects he doesn’t understand or a bunch of straightforward badges on today’s most used platform for data science?