UC Berkeley · FSDL | Full Stack Deep Learning (2021)

FSDL (2021)· 课程资料包 @ShowMeAI

视频
中英双语字幕

课件
一键打包下载

笔记
官方笔记翻译

代码
作业项目解析

视频 · B 站 [扫码或点击链接]
https://www.bilibili.com/video/BV1iL411t7jE

课件 & 代码 · 博客 [扫码或点击链接]
http://blog.showmeai.tech/berkeley-fsdl

深度学习 可解释性 持续集成 迁移学习 卷积神经网络 神经网络 计算机视觉 Transformer 深度神经网络调试 循环神经网络 数据管理 部署模型 监控模型 测试

Awesome AI Courses Notes Cheatsheets 是 ShowMeAI 资料库的分支系列，覆盖最具知名度的 TOP50+ 门 AI 课程，旨在为读者和学习者提供一整套高品质中文学习笔记和速查表。

点击课程名称，跳转至课程**资料包**页面，**一键下载**课程全部资料！

机器学习	深度学习	自然语言处理	计算机视觉
Stanford · CS229	Stanford · CS230	Stanford · CS224n	Stanford · CS231n
# Awesome AI Courses Notes Cheatsheets· 持续更新中			
知识图谱	图机器学习	深度强化学习	自动驾驶
Stanford · CS520	Stanford · CS224W	UCBerkeley · CS285	MIT · 6.S094

微信公众号

资料下载方式 2：扫码点击底部菜单栏称为 **AI 内容创作者？** 回复 [添砖加瓦]

FSDL Lecture 13: Machine Learning Teams

Video and Slides by Josh Tobin - posted on the FSDL Course Website
Notes were taken by James Le and Vishnu Rachakonda

Over the past few years, machine learning (ML) has grown tremendously. But as young as ML is as a discipline, the craft of managing an ML team is even younger. Many of today's ML managers were thrust into management roles out of necessity or because they were the best individual contributors, and many come from purely academic backgrounds. At some companies, engineering or product leaders are tasked with building new ML functions without real ML experience.

Running any technical team is hard:

- You have to hire great people.
- You need to manage and develop them.
- You need to manage your team's output and make sure your vectors are aligned.
- You would want to make good long-term technical choices and manage technical debt.
- You also must manage expectations from leadership.

Running an ML team is even harder:

- ML talents are expensive and scarce.
- ML teams have a diverse set of roles.
- ML projects have unclear timelines and high uncertainty.
- ML is also the "high-interest credit card of technical debt."
- Leadership often doesn't understand ML.

The goals of this lecture are two-fold: (1) to give you insight into how to think about building and managing ML teams (as a leader); and (2) to help you get a job in ML (as a newcomer).

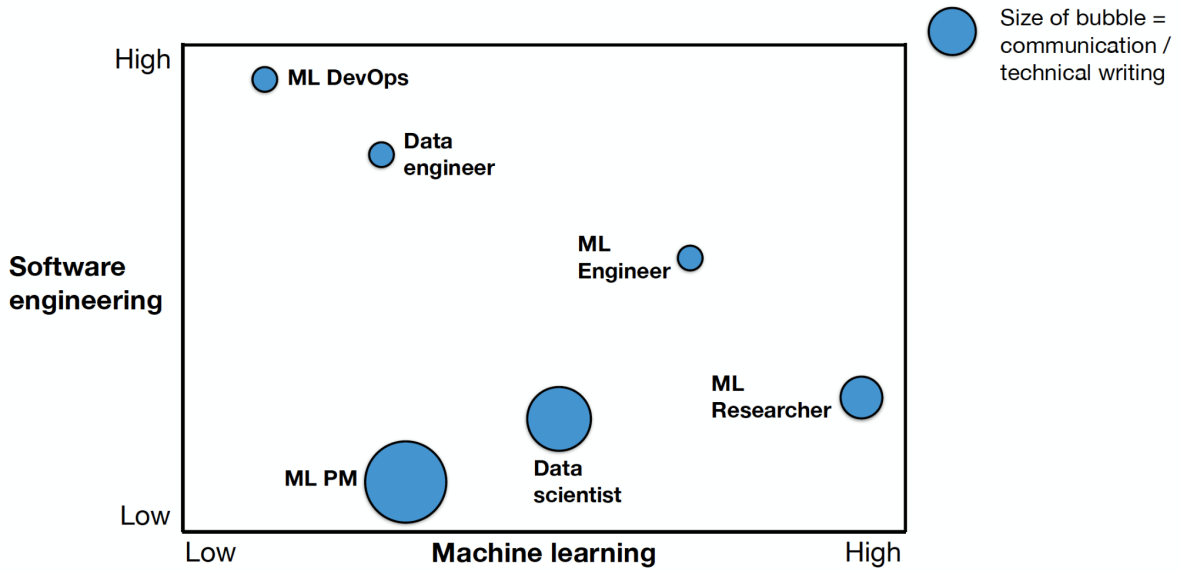
1 - ML Roles

Common Roles

Let's take a look at the most common ML roles and the skills they require:

- The **ML Product Manager** works with the ML team, other business functions, the end-users, and the data owners. This person designs documentation, creates wireframes, and comes up with the plan to prioritize and execute ML projects.
- The **DevOps Engineer** deploys and monitors production systems. This person handles the infrastructure that runs the deployed ML product using platforms like AWS or GCP.
- The **Data Engineer** builds data pipelines, aggregates and collects data from storage, and monitors data behavior. This person works with distributed systems using tools such as Hadoop, Kafka, Airflow.
- The **ML Engineer** trains and deploys prediction models. This person uses tools like TensorFlow and Docker to work with prediction systems running on real data in production.
- The **ML Researcher** trains prediction models, often those that are forward-looking or not production-critical. This person uses libraries like TensorFlow and PyTorch on notebook environments to build models and reports describing their experiments.
- The **Data Scientist** is a blanket term used to describe all of the roles above. In some organizations, this role entails answering business questions via analytics. He/she can work with wide-ranging tools from SQL and Excel to Pandas and Scikit-Learn.

Skills Required



So what skills are needed for these roles? The chart above displays a nice visual, where the horizontal axis is the level of ML expertise and the size of the bubble is the level of communication and technical writing (the bigger, the better).

- The **ML DevOps** is primarily a software engineering role, which often comes from a standard software engineering pipeline.
- The **Data Engineer** belongs to the software engineering team that works actively with ML teams.
- The **ML Engineer** requires a rare mix of ML and Software Engineering skills. This person is either an engineer with significant self-teaching OR a science/engineering Ph.D. who works as a traditional software engineer after graduate school.
- The **ML Researcher** is an ML expert who usually has an MS or Ph.D. degree in Computer Science or Statistics or finishes an industrial fellowship program.
- The **ML Product Manager** is just like a traditional Product Manager, but with a deep knowledge of the ML development process and mindset.
- The **Data Scientist** role constitutes a wide range of backgrounds from undergraduate to Ph.D. students.

2 - ML Organizations

Organization Archetypes

There exists not yet a consensus on the right way to structure an ML team. Still, a few best practices are contingent upon different organization archetypes and their ML maturity level. First, let's see what the different ML organization archetypes are.

Archetype 1 - Nascent and Ad-hoc ML

- These are organizations where no one is doing ML, or ML is done on an ad-hoc basis. Obviously, there is little ML expertise in-house.
- They are either small-to-medium businesses or less technology-forward large companies in industries like education or logistics.
- There is often low-hanging fruit for ML.
- But there is little support for ML projects, and it's challenging to hire and retain good talent.

Archetype 2 - Research and Development ML

- These are organizations in which ML efforts are centered in the R&D arm of the organization. They often hire ML researchers and doctorate students with experience publishing papers.
- They are larger companies in sectors such as oil and gas, manufacturing, or telecommunications.
- They can hire experienced researchers and work on long-term business priorities to get big wins.
- However, it is very difficult to get quality data. Most often, this type of research work rarely translates into actual business value, so usually, the amount of investment remains small.

Archetype 3 - Product-Embedded ML

- These are organizations where certain product teams or business units have ML expertise alongside their software or analytics talent. These ML individuals report up to the team's engineering/tech lead.
- They are either software companies or financial services companies.
- ML improvements are likely to lead to business value. Furthermore, there is a tight feedback cycle between idea iteration and product improvement.
- Unfortunately, it is still very hard to hire and develop top talent, and access to data and compute resources can lag. There are also potential conflicts between ML project cycles and engineering management, so long-term ML projects can be hard to justify.

Archetype 4 - Independent ML Division

- These are organizations in which the ML division reports directly to senior leadership. The ML Product Managers work with Researchers and Engineers to build ML into client-facing products. They can sometimes publish long-term research.
- They are often large financial services companies.
- Talent density allows them to hire and train top practitioners. Senior leaders can marshal data and compute resources. This gives the organizations to invest in tooling, practices, and culture around ML development.
- A disadvantage is that model handoffs to different business lines can be challenging since users need the buy-in to ML benefits and get educated on the model use. Also, feedback cycles can be slow.

Archetype 5 - ML-First

- These are organizations in which the CEO invests in ML, and there are experts across the business focusing on quick wins. The ML division works on challenging and long-term projects.
- They are large tech companies and ML-focused startups.
- They have the best data access (data thinking permeates the organization), the most attractive recruiting funnel (challenging ML problems tends to attract top talent), and the easiest deployment procedure (product teams understand ML well enough).
- This type of organization archetype is hard to implement in practice since it is culturally difficult to embed ML thinking everywhere.

Team Structure Design Choices

Depending on the above archetype that your organization resembles, you can make the appropriate design choices, which broadly speaking follow these three categories:

1. **Software Engineer vs. Research**: To what extent is the ML team responsible for building or integrating with software? How important are Software Engineering skills on the team?
2. **Data Ownership**: How much control does the ML team have over data collection, warehousing, labeling, and pipelining?
3. **Model Ownership**: Is the ML team responsible for deploying models into production? Who maintains the deployed models?

Below are our design suggestions:

If your organization focuses on **ML R&D**:

- Research is most definitely prioritized over Software Engineering skills. Because of this, there would potentially be a lack of collaboration between these two groups.

- ML team has no control over the data and typically will not have data engineers to support them.
- ML models are rarely deployed into production.

If your organization has **ML embedded into the product**:

- Software Engineering skills will be prioritized over Research skills. Often, the researchers would need strong engineering skills since everyone would be expected to product-ionize his/her models.
- ML teams generally do not own data production and data management. They will need to work with data engineers to build data pipelines.
- ML engineers totally own the models that they deploy into production.

If your organization has **an independent ML division**:

- Each team has a potent mix of engineering and research skills; therefore, they work closely together within teams.
- ML team has a voice in data governance discussions, as well as a robust data engineering function.
- ML team hands-off models to users but is still responsible for maintaining them.

If your organization is **ML-First**:

- Different teams are more or less research-oriented, but in general, research teams collaborate closely with engineering teams.
- ML team often owns the company-wide data infrastructure.
- ML team hands the models to users, who are responsible for operating and maintaining them.

The picture below neatly sums up these suggestions:

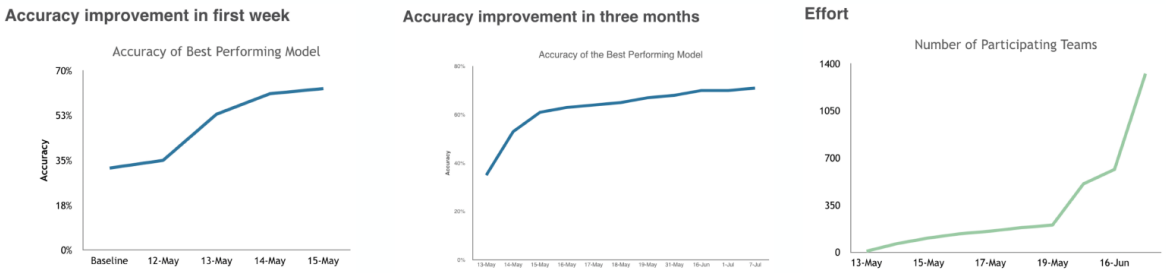
	ML R&D	Embedded ML	ML Function	ML First
Software engineering vs research	• Research prioritized over SWE skills • Researcher-SWE collaboration lacking	• SWE skills prioritized over research skills • Often, all researchers need strong SWE as everyone expected to deploy	• Each team has a strong mix of SWE and research skills • SWE and researchers work closely together within team	• Different teams are more or less research oriented • Research teams collaborate closely with SWE teams
Data ownership	• ML team has no control over data • ML team typically will not have data engineering component	• ML team generally does not own data production / mgmt • Work with data engineers to build pipelines	• ML team has a voice in data governance discussions • ML team has strong internal data engineering function	• ML team often owns company-wide data infrastructure
Model ownership	• Models are rarely deployed into production	• ML engineers own the models that they deploy into production	• ML team hands off models to user, but is responsible for maintaining them	• ML team hands off models to user, who operates and maintains them

3 - Managing ML Teams

Managing ML Teams Is Challenging

The process of actually managing an ML team is quite challenging for four reasons:

1. **Engineering Estimation:** It's hard to know how easy or hard an ML project is in advance. As you explore the data and experiment with different models, there is enormous scope for new learnings about the problem that materially impact the timeline. Furthermore, knowing what methods will work is often impossible. This makes it hard to say upfront how long or how much work may go into an ML project.
2. **Nonlinear Progress:** As the chart below from a blog post by Lukas Biewald (CEO of Weights and Biases) shows, progress on ML projects is unpredictable over time, even when the effort expended grows considerably. It's very common for projects to stall for extended periods of time.



3. **Cultural gaps:** The relative culture of engineering and research professionals is very different. Research tends to favor novel, creative ideas, while engineering prefers tried and true methods that work. As a result, ML teams often experience a clash of cultures, which can turn toxic if not appropriately managed. A core challenge of running ML teams is addressing the cultural barriers between ML and software engineering so that teams can harmoniously experiment and deliver ML products.
4. **Leadership Deficits**: It's common to see a lack of detailed understanding of ML at senior levels of management in many companies. As a result, expressing feasibility and setting the right expectations for ML projects, especially high-priority ones, can be hard.

How To Manage ML Teams Better

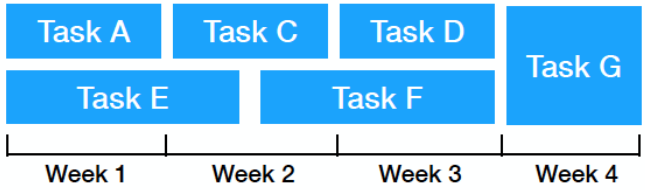
Managing ML teams is hardly a solved problem, but you can take steps to improve the process.

Plan Probabilistically

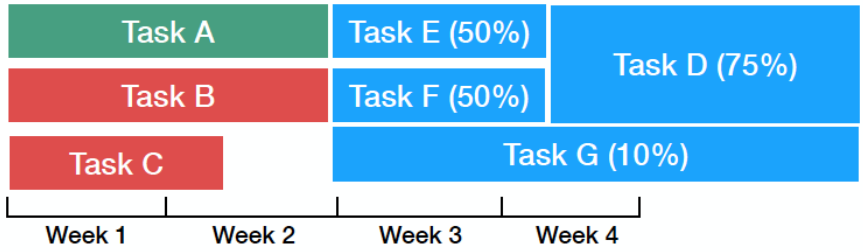
Many engineering projects are managed in a waterfall fashion, with the sequential tasks defined up front clearly. Instead of forcing this method of engineering management on difficult ML projects, try assigning a likelihood of success to different tasks to better capture the

experimental process inherent to ML engineering. As these tasks progress or stall, rapidly re-evaluate your task ordering to better match what is working. Having this sense of both (1) **how likely a task is to succeed** and (2) **how important it is** makes project planning considerably more realistic.

From:



To:



Have A Portfolio Of Approaches

Embrace multiple ideas and approaches to solve crucial research challenges that gate production ML. Don't make your plan dependent on one approach working!

Measure Inputs, Not Results

As you work through several approaches in your portfolio, do not overly emphasize whose ideas ultimately work as a reflection of contribution quality. This can negatively impact team members' creativity, as they focus more on trying to find only what they currently think could work, rather than experimenting in a high-quality fashion (which is ultimately what leads to ML success).

Have Researchers and Engineers Work Together

The collaboration between engineering and research is essential for quality ML products to get into production. Emphasize collaboration across the groups and professionals!

Get End-to-end Pipelines Together Quickly to Demonstrate Quick Wins

Taking this approach makes it more likely that your ML project will succeed in the long term. It allows you to demonstrate progress to your leadership more effectively and clearly.

Educate Leadership on ML Timeline Uncertainty

This can be hard, as leadership is ultimately accountable for addressing blind spots and understanding timeline risk. There are things you can do, however, to help improve leadership's knowledge about ML timelines. Avoid building hype around narrow progress metrics material only to the ML team (e.g., "*We improved F1 score by 0.2 and have achieved awesome performance!*"). Instead, be realistic, communicate risk, and emphasize real product impact (e.g., "Our model improvements should increase the number of conversions by 10%, though we must continue to validate its performance on additional demographic factors.) Sharing resources like this a16z primer and this class from Prof. Pieter Abbeel can increase awareness of your company's leadership.

4 - Hiring/Getting Hired

The AI Talent Gap

With the novelty of ML systems, it's fair to say that not many people have built real ML systems. Estimates vary from as few as 10,000 (Element AI) to as many as 200-300,000 people (Tencent). Whatever way you slice the numbers (contained in this blog post), the reality is that there is not much-experienced talent in the AI/ML field, especially compared to the number of trained software developers in the US (3.6M) or in the world (18.2M).

Sourcing

Because of this shallow talent pool and the skyrocketing demand, hiring for ML positions is pretty hard. Typical ML roles come in the following structure:

- ML Adjacent roles: ML product manager, DevOps, Data Engineer
- Core ML Roles: ML Engineer, ML Research/ML Scientist
- Business analytics roles: Data Scientist

For ML adjacent roles, traditional ML knowledge is less important, as demonstrated interest, conversational understanding, and experience can help these professionals play an impactful role on ML teams. Let's focus on how to hire for **the core ML roles**.

While there's no perfect way to **hire ML engineers**, there's definitely a wrong way to hire them, with extensive job descriptions that demand only the best qualifications. Certainly, there are many good examples of this bad practice floating around.

- Rather than this unrealistic process, **consider hiring for software engineering skills, an interest in ML, and a desire to learn**. You can always train people in the art and science of ML, especially when they come with strong software engineering fundamentals.

- Another option is to consider adding **junior talent**, as many recent grads come out with good ML knowledge nowadays.
- Finally, and most importantly, **be more specific** about what you need the position and professional to do. It's impossible to find one person that can do everything from full-fledged DevOps to algorithm development.

To **hire ML researchers**, here are our tips:

- Evaluate the quality of publications, over the quantity, with an eye towards the originality of the ideas, the execution, etc.
- Prioritize researchers that focus on important problems instead of trendy problems.
- Experience outside academia is also a positive, as these researchers may be able to transition to industry more effectively.
- Finally, keep an open mind about research talent and consider talented people without PhDs or from adjacent fields like physics, statistics, etc.

To find quality candidates for these roles, some ideas for **sourcing** are:

- To experiment with standard job recruiting avenues like LinkedIn, Hired, recruiters, on-campus-recruiting, etc.
- To monitor arXiv and top conferences and flag first authors of papers you like.
- To look for good implementations of papers you like.
- To attend ML research conferences (NeurIPS, ICML, etc.)

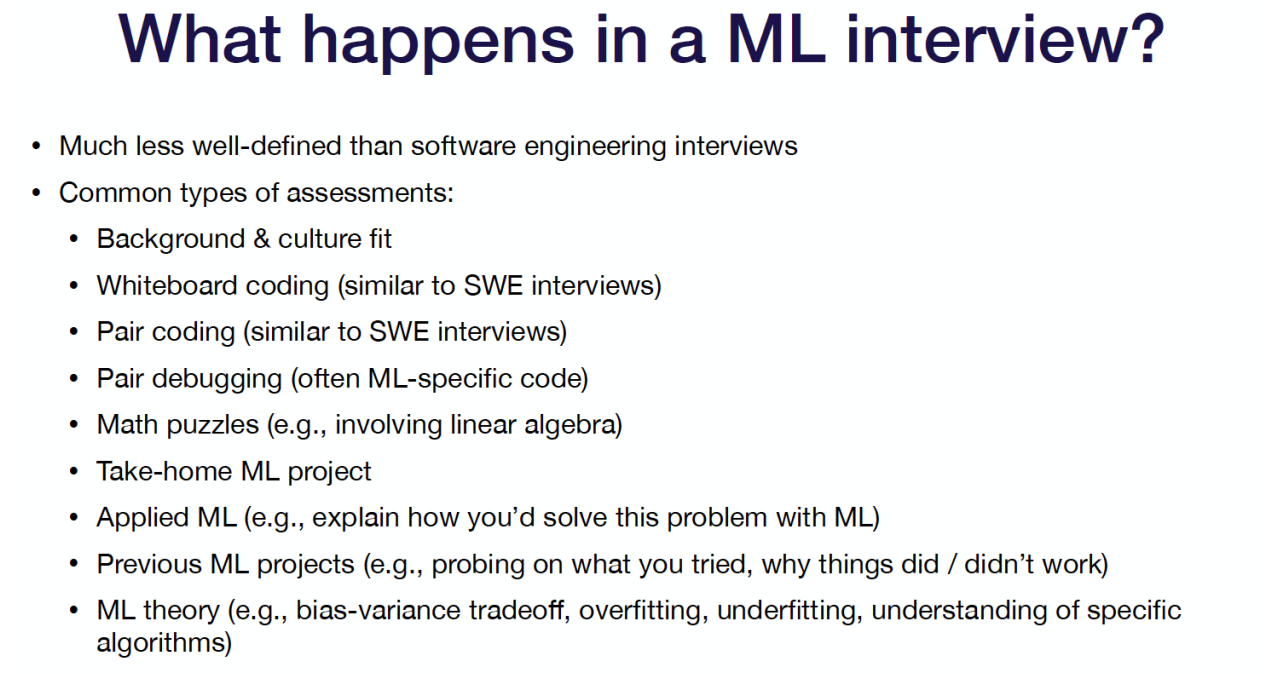
As you seek to recruit, stay on top of what professionals want and make an effort to position your company accordingly. ML practitioners want to be empowered to do great work with interesting data. Building a culture of learning and impact can help recruit the best talent to your team. Additionally, sell sell sell! Talent needs to know how good your team is and how meaningful the mission can be.

What do machine learning practitioners want?	How to make your company stand out?
• Work with cutting edge tools & techniques	• Work on research-oriented projects. Publicize them. Invest in tooling for your team & empower employees to try new tools.
• Build skills / knowledge in an exciting field	• Build team culture around learning (reading groups, learning days, professional development budget, conference budget)
• Work with excellent people	• Hire high-profile people. Help your best people build their profile through publishing blogs & papers.
• Work on interesting datasets	• Sell the uniqueness of your dataset in recruiting materials.
• Do work that matters	• Sell the mission of your company and potential impact of machine learning on that mission. Work on projects that have a tangible impact today.

Interviewing

As you interview candidates for ML roles, try to **validate your hypotheses of their strengths while testing a minimum bar on weaker aspects**. For example, make sure ML researchers can think creatively about new ML problems while ensuring they meet a baseline for code quality. It's essential to test both ML knowledge and software engineering skill for all industry professionals, though the relative strengths can vary.

The actual ML interview process is much less well-defined than software engineering interviews, though it is modeled off of it. Some helpful inclusions are projects or exercises that test the ability to work with ML-specific code, like take-home ML projects.



Finding A Job

To find an ML job, you can take a look at the following sources:

- Standard sources such as LinkedIn, recruiters, on-campus recruiting, etc.
- ML research conferences (NeurIPS, ICLR, ICML).
- Apply directly (remember, there's a talent gap!).

Standing out for competitive roles can be tricky! Here are some tips in increasing order of impressiveness that you can apply to differentiate yourself:

1. Build software engineering skills (e.g., at a well-known software company).
2. Exhibit ML interest (e.g., conference attendance, online courses certificates, etc.).

3. Show you have a broad knowledge of ML (e.g., write blog posts synthesizing a research area).
4. Demonstrate ability to get ML projects done (e.g., create side projects, re-implement papers).
5. Prove you can think creatively in ML (e.g., win Kaggle competitions, publish papers).

As you prepare for interviews, prepare for both the traditional ML theoretical topics and the general software engineering interview (e.g., read Cracking the Coding Interview).

5 - Conclusion

Being a new and evolving discipline for most traditional organizations, forming ML teams is full of known and unknown challenges. Here are the final few take-homes:

- There are **many different skills** involved in production ML, so there are opportunities for many people to contribute.
- ML teams are becoming more standalone and more **interdisciplinary**.
- Managing ML teams is complex. There is no silver bullet, but shifting toward **probabilistic planning** can help.
- **ML talent is scarce**. As a manager, be specific about what skills are must-have in the ML job descriptions. As a job seeker, it can be brutally challenging to break in as an outsider, so use projects as a signal to build awareness.

UC Berkeley · FSDL | Full Stack Deep Learning (2021)

FSDL (2021)· 课程资料包 @ShowMeAI

视频
中英双语字幕

课件
一键打包下载

笔记
官方笔记翻译

代码
作业项目解析

视频 · B 站 [扫码或点击链接]
https://www.bilibili.com/video/BV1iL411t7jE

课件 & 代码 · 博客 [扫码或点击链接]
http://blog.showmeai.tech/berkeley-fsdl

深度学习 可解释性 持续集成 迁移学习 卷积神经网络 神经网络 计算机视觉 Transformer 深度神经网络调试 循环神经网络 数据管理 部署模型 监控模型 测试

Awesome AI Courses Notes Cheatsheets 是 ShowMeAI 资料库的分支系列，覆盖最具知名度的 TOP50+ 门 AI 课程，旨在为读者和学习者提供一整套高品质中文学习笔记和速查表。

点击课程名称，跳转至课程**资料包**页面，**一键下载**课程全部资料！

机器学习	深度学习	自然语言处理	计算机视觉
Stanford · CS229	Stanford · CS230	Stanford · CS224n	Stanford · CS231n
# Awesome AI Courses Notes Cheatsheets· 持续更新中			
知识图谱	图机器学习	深度强化学习	自动驾驶
Stanford · CS520	Stanford · CS224W	UCBerkeley · CS285	MIT · 6.S094

微信公众号

资料下载方式 2：扫码点击底部菜单栏称为 **AI 内容创作者？** 回复 [添砖加瓦]