TILL LOHFINK & STEFAN ZAPF

# CORE ML BOOTCAMP: BUILD IOS & MAC APPS WITH DEEP LEARNING

Use Core ML to add intelligence to your application

# PART 1

Our Bootcamp

What can AI do for you?

The Create ML Tool

CoreML Conversion

# CORE ML BOOTCAMP

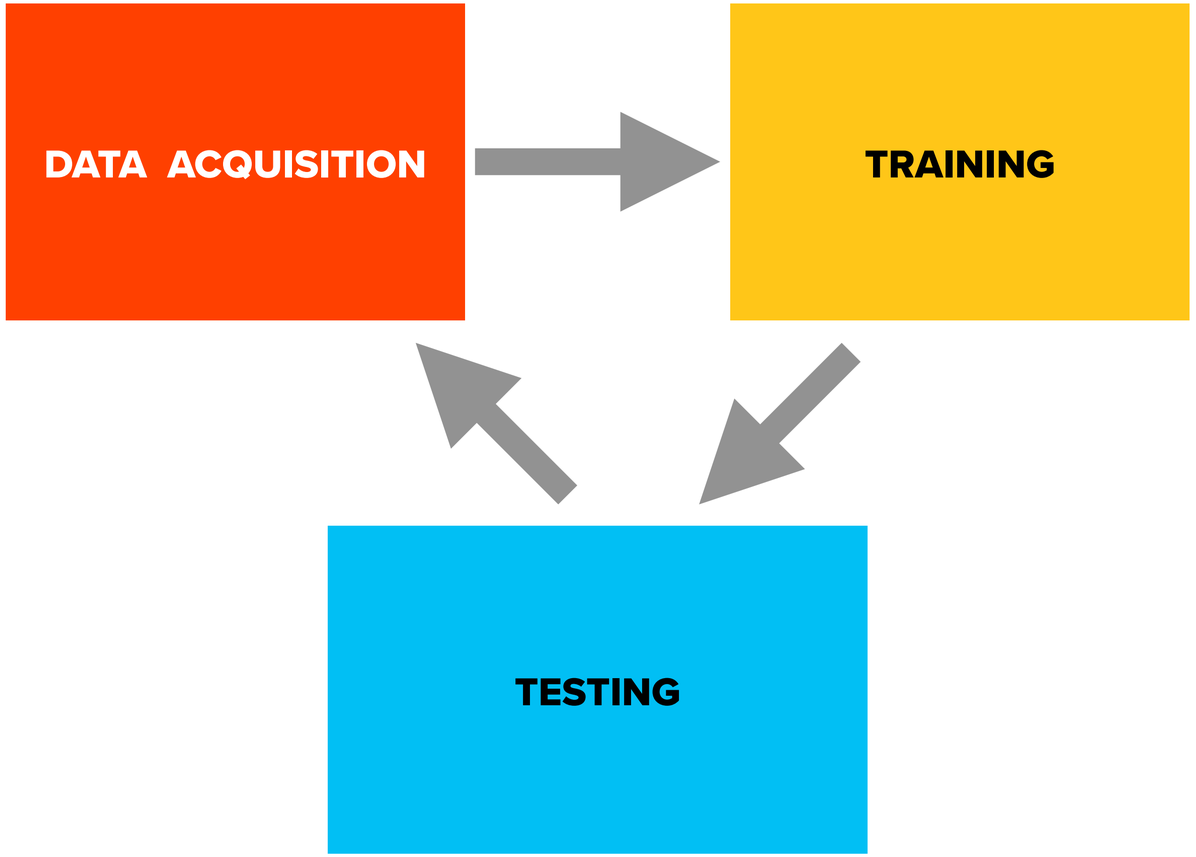
# DEEP LEARNING IS MACHINE LEARNING WITH AUTOMATED FEATURE ENGINEERING

# BEFORE & AFTER DL 1

- Before Picture
- Object Detection:
  - Edge Detection
  - Template Matching
- Each method had problems:
  - Lighting Conditions
  - Different Sizes / Angles
  - Weather Conditions such as snow / rain ...
- A lot of research on all the sub problems
- Careers were mad in researching these sub problems

# BEFORE & AFTER DL 1

- Today with Deep Learning
- Data, Training, Testing - Triad
- If lightning is a problem
  - Find new data with different lightning
  - Train again
  - Test again
  - Repeat



# FOUR LEVELS OF AI ENGINEERING

# FOUR LEVELS OF AI ENGINEERING

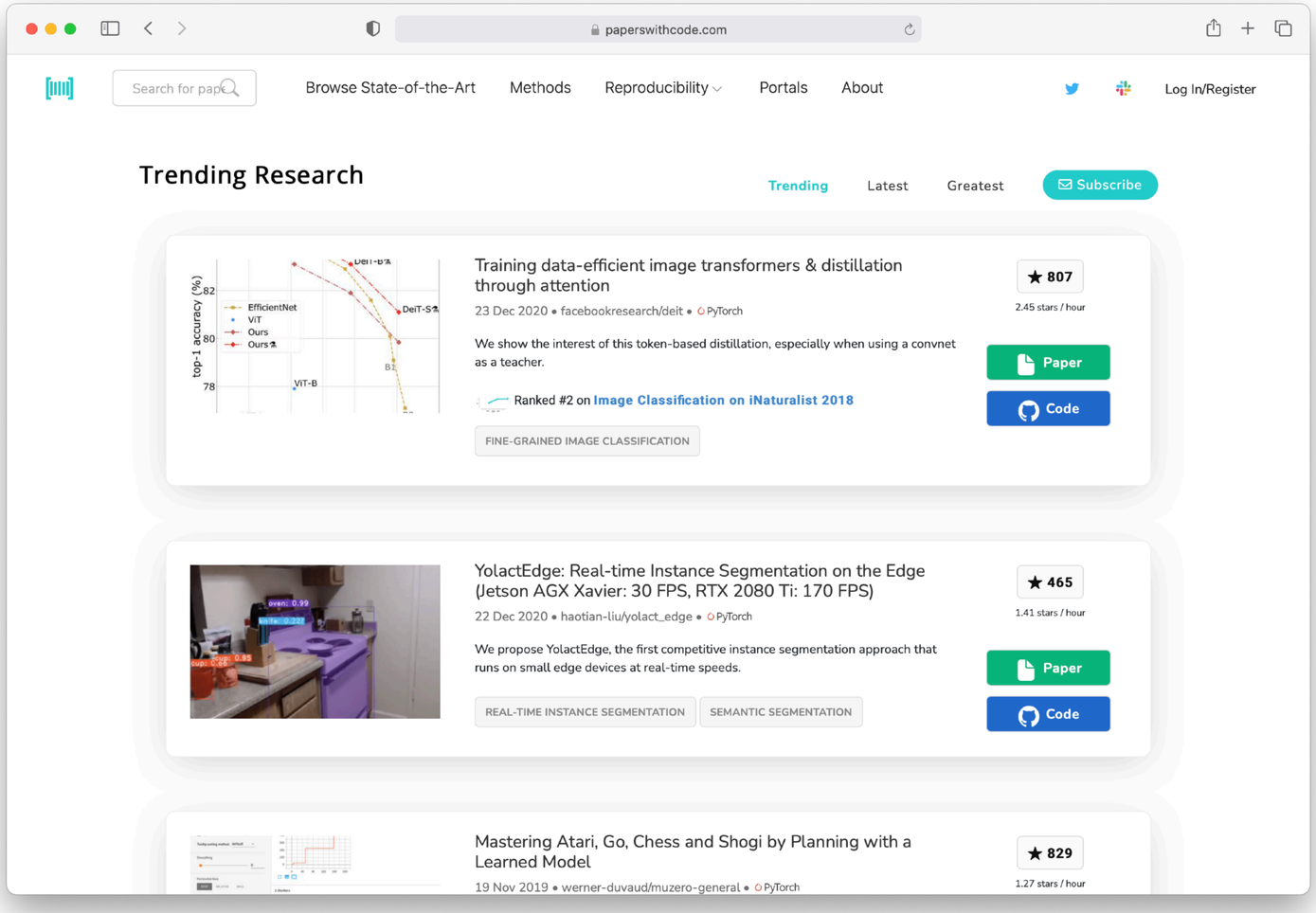
- 1 Using public APIs
  - e. g. Azure Cognitive Services, OpenAI, etc.
- 2 Utilize Transfer Learning with existing state-of-the-art solutions
  - Like in this class
  - Build on top of leading architectures like BERT or ResNet
- 3 Coming up with your own models
  - Work directly in PyTorch or Tensorflow
  - Integrate State-Of-The-Art research
- 4 Auto ML
  - Create AIs that can create AIs
  - Deep Reinforcement Learning

# AI TASKS: WHAT CAN AI DO FOR YOU?

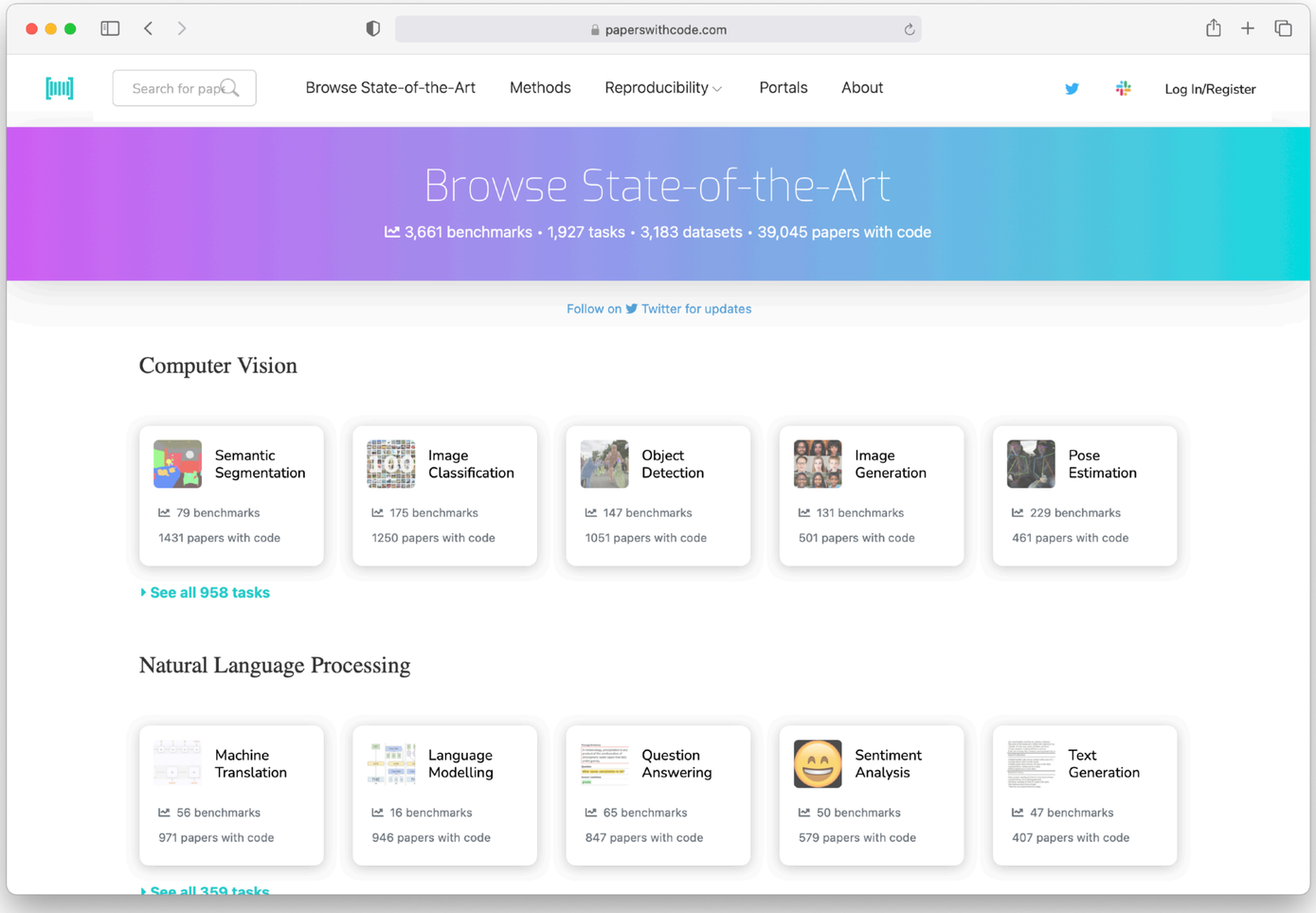
# WHAT CAN AI DO FOR YOU?

- Computer Vision
  - Semantic Segmentation
  - Image classification
  - Object Detection
  - Image Generation
  - Pose Estimation
  - Super-Resolution
  - Object Tracking
  - Activity Recognition
- Natural Language Processing
  - Machine Translation
  - Language Modeling
  - Question Answering
  - Sentiment Analysis
  - Text Generation
  - Named Entity Recognition
  - Text Summarization
  - Semantic Textual Similarity
  - Part-Of-Speech Tagging

# SOLVE THE MOST DIFFICULT PROBLEMS TO DATE WITH EASE



# SOLVE THE MOST DIFFICULT PROBLEMS TO DATE WITH EASE



# HOW CAN YOU USE THESE AI CAPABAILITIES IN YOUR APP?

# DEMO TIME: CREATE ML TOOL

# VISION FRAMEWORK

- Out-Of-The-Box components to build image and video processing applications
- Quite extensive; offers solutions for image classification, similarity analysis and object detection with little code
- Great for moving objects featuring object tracking and even trajectory detection
- Neat stuff like text detection, lots of health related algorithms like pose detection etc.

- In general works like this: you bring an image and pass it to an instance of an VNImageRequestHandler which can call an instance of the desired VNSomethingSomethingRequest
- We will use this framework later!

# NATURAL LANGUAGE FRAMEWORK

- Out-Of-The-Box components to build text-based applications - to be more specific
- Not as extensive as Apple's Vision Framework
- Brings some great tools for tokenization and lemmatization (finding a word's stem; much more interesting in romance languages)
- Can help you with a lot of tagging feature like named entity recognition or part-of-speed tagging

- Very easy to implement in your application: Instantiate the desired class and call a method passing your text
- If you want to do more sophisticated things like sentiment analysis, Q & As etc. you have to bring your own models / implementations (i. e. Papers With Code)

# WHAT CAN APPLE DO FOR YOU? (OUT OF THE BOX)

- Computer Vision
  - Semantic Segmentation
  - Image classification ✅
  - Object Detection ✅
  - Image Generation
  - Image Similarity Analysis ✅
  - Pose Estimation ✅
  - Super-Resolution
  - Object Tracking ✅
  - Activity Recognition ✅
- Natural Language Processing
  - Machine Translation
  - Language Modeling
  - Question Answering
  - Sentiment Analysis
  - Text Generation
  - Named Entity Recognition
  - Text Summarization
  - Semantic Textual Similarity
  - Part-Of-Speech Tagging

# WHAT CAN APPLE DO FOR YOU? (OUT OF THE BOX)

- Computer Vision
  - Semantic Segmentation
  - Image classification ✅
  - Object Detection ✅
  - Image Generation
  - Image Similarity Analysis ✅
  - Pose Estimation ✅
  - Super-Resolution
  - Object Tracking ✅
  - Activity Recognition ✅
- Natural Language Processing
  - Machine Translation
  - Language Modeling
  - Question Answering
  - Sentiment Analysis ✅
  - Text Generation
  - Named Entity Recognition ✅
  - Text Summarization
  - Semantic Textual Similarity
  - Part-Of-Speech Tagging ✅

# IMPORTING EXISTING MODELS OUTSIDE OF APPLE'S ECOSYSTEM

HUGGINGFACE

# DEMO TIME: GOOGLE COLAB

# INTERMISSION

# PART 2: THE LEGAL APP

App Specifications

The Transformer & Bert

Coding 1: The UI

Coding 2: The Pdf Component

Coding 3: Using BERT with GCD

# THE LEGAL APP

# LEGAL APP PURPOSE

- A legal argument is based on:
  - facts, and
  - legal argument, mostly applying or distinguishing precedent
- When presenting a case, the lawyer will need to find the facts in a large corpus of documents that support their legal argument and build defenses against facts that do not confirm the legal narrative
- During discovery, the pertinent fact can be buried in a large corpus of documents
- Large Law firms have a staff of paralegals and other support staff to find the needle in a haystack
- Smaller law firms are more affordable, but often don’t have the staff to tackle a large corpus
- Lawyers can ask questions about facts to the legal app to find the right facts and thereby leveling the playing field

**YOUR AI HELPS DEMOCRATIZE LEGAL REPRESENTATION**

# THE TRANSFORMER

# Attention Is All You Need


**Ashish Vaswani*** Google Brain avaswani@google.com

**Noam Shazeer*** Google Brain noam@google.com

**Niki Parmar*** Google Research nikip@google.com

**Jakob Uszkoreit*** Google Research usz@google.com

**Llion Jones*** Google Research llion@google.com

**Aidan N. Gomez*** † University of Toronto aidan@cs.toronto.edu

**Łukasz Kaiser*** Google Brain lukaszkaiser@google.com

**Illia Polosukhin*** ‡ illia.polosukhin@gmail.com


**ORIGIN PAPER FOR TRANSFORMERS**

**BERT IS A SPECIFIC TRANSFORMER**

**Attention is all you need**

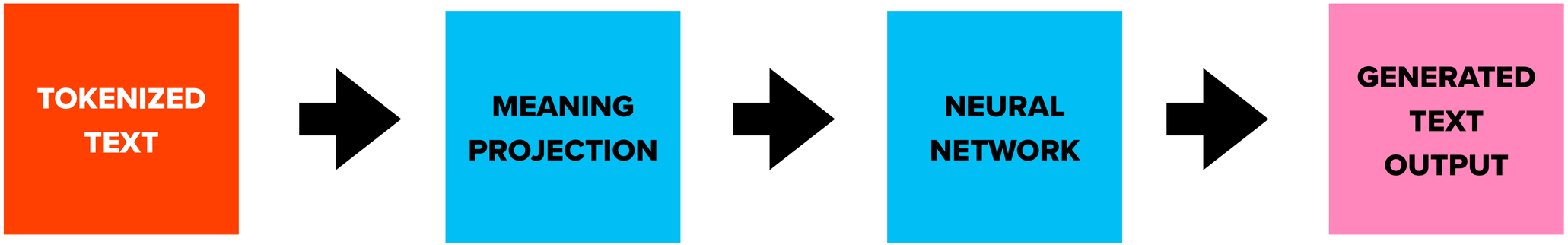
A Vaswani, N Shazeer, N Parmar… - Advances in neural …, **2017** - papers.nips.cc

The dominant sequence transduction models are based on complex recurrent orconvolutional neural networks in an encoder and decoder configuration. The best performing such models also connect the encoder and decoder through an attentionm …

☆ Cited by 16137 Related articles All 20 versions


*Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems. 2017.*

# BIRDS EYE'S VIEW OF NATURAL LANGUAGE UNDERSTANDING




Unsupervised Pre-Trained Task

- Masked Language Modelling
- Next Sentence Prediction
- Causal or Bi-Directional

PRE-TRAINED TO UNDERSTAND A SET OF LANGUAGE(S)

TRAINED ON DOWNSTREAM TASK

Supervised Downstream Task

- Q&A
- Neural Machine Translation
- Chat Bot
- ... Anything you train it to do

# THE EVOLUTION OF MEANING PROJECTION

- Word Embeddings
- Transformers

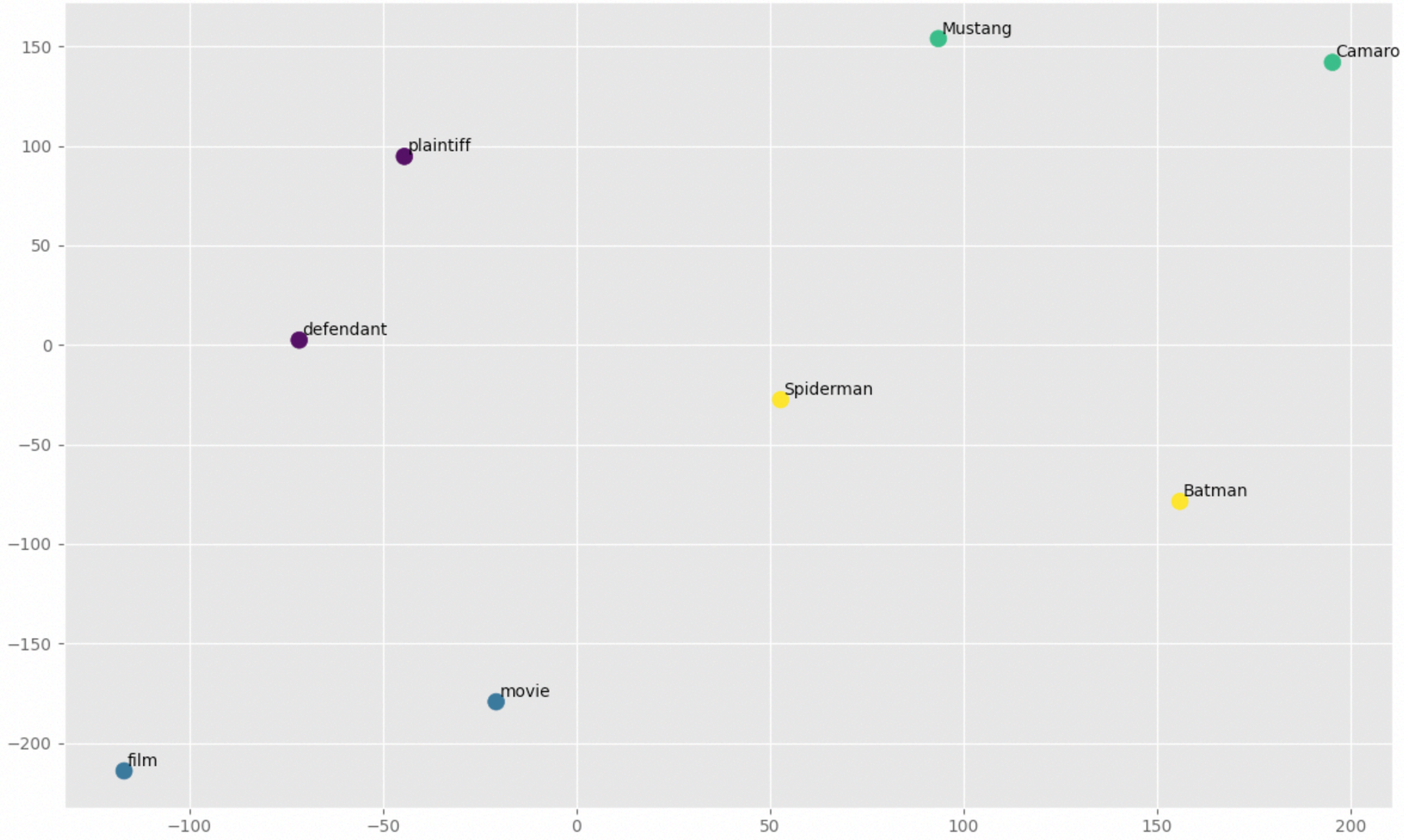
# WORD EMBEDDINGS, MEANING AND LATENT SPACE

**LATENT SPACE OF MEANING**

**EVERY WORD HAS A PLACE IN THE LATENT SPACE**

**WORDS THAT HAVE A HIGH SEMANTIC SIMILARITY ARE CLOSER IN THE LATENT SPACE**

**METHOD: WORD EMBEDDING WORD -> VECTOR**



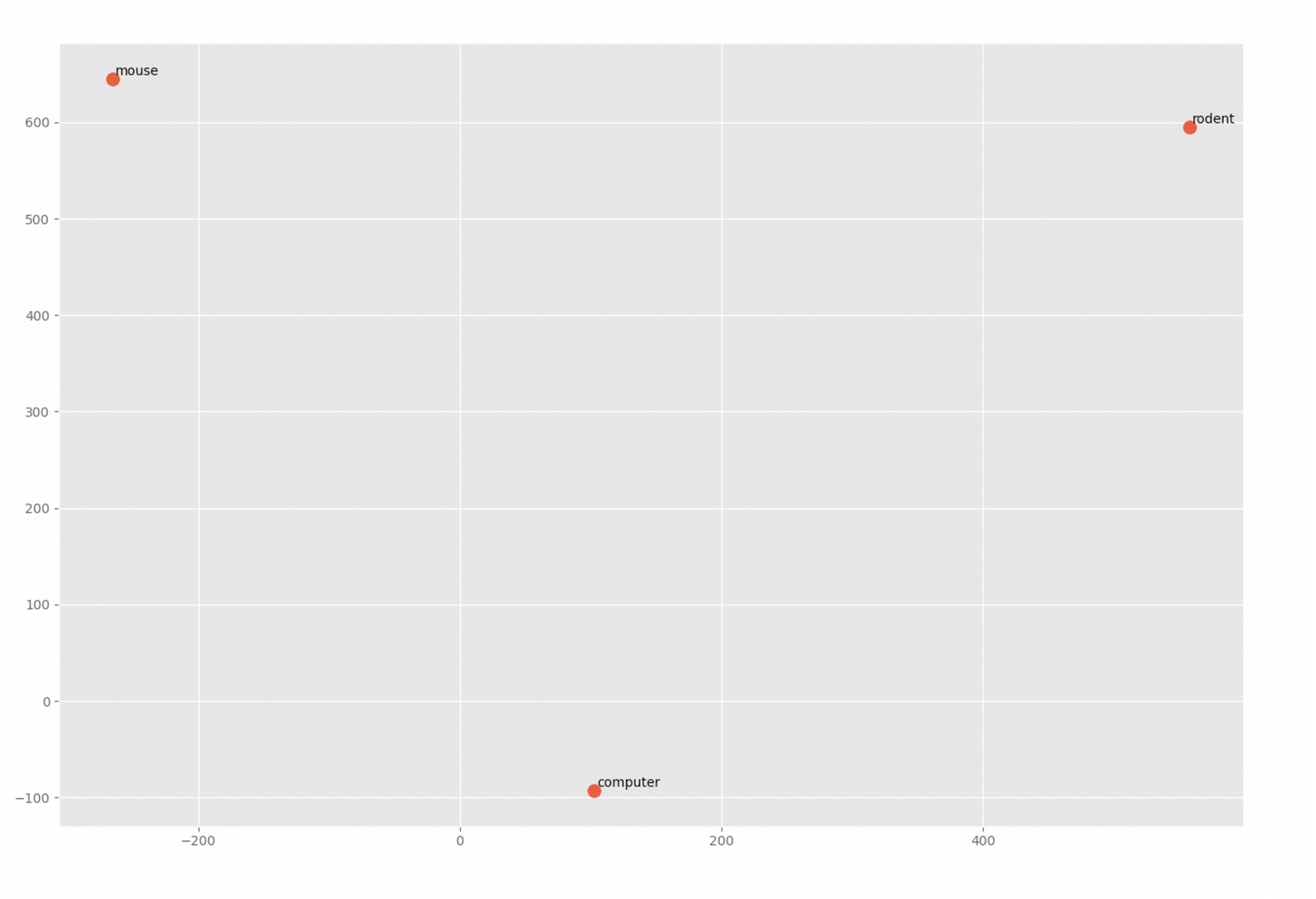
# CHAT STORMING: WHAT COULD BE A PROBLEM WITH WORD EMBEDDINGS?
# PLEASE WRITE YOUR IDEAS IN THE CHAT

# AMBIGUITIES

**SYMBOL “MOUSE”**

**POINTS TO TWO MEANINGS**

**AMBIGUOUS CLOSENESS IN LATENT SPACE OF MEANING**



# POLL: HOW CAN YOU DISAMBIGUATE THE DIFFERENT MEANINGS ‘THE ARTISTS DRAWS A GUN’?

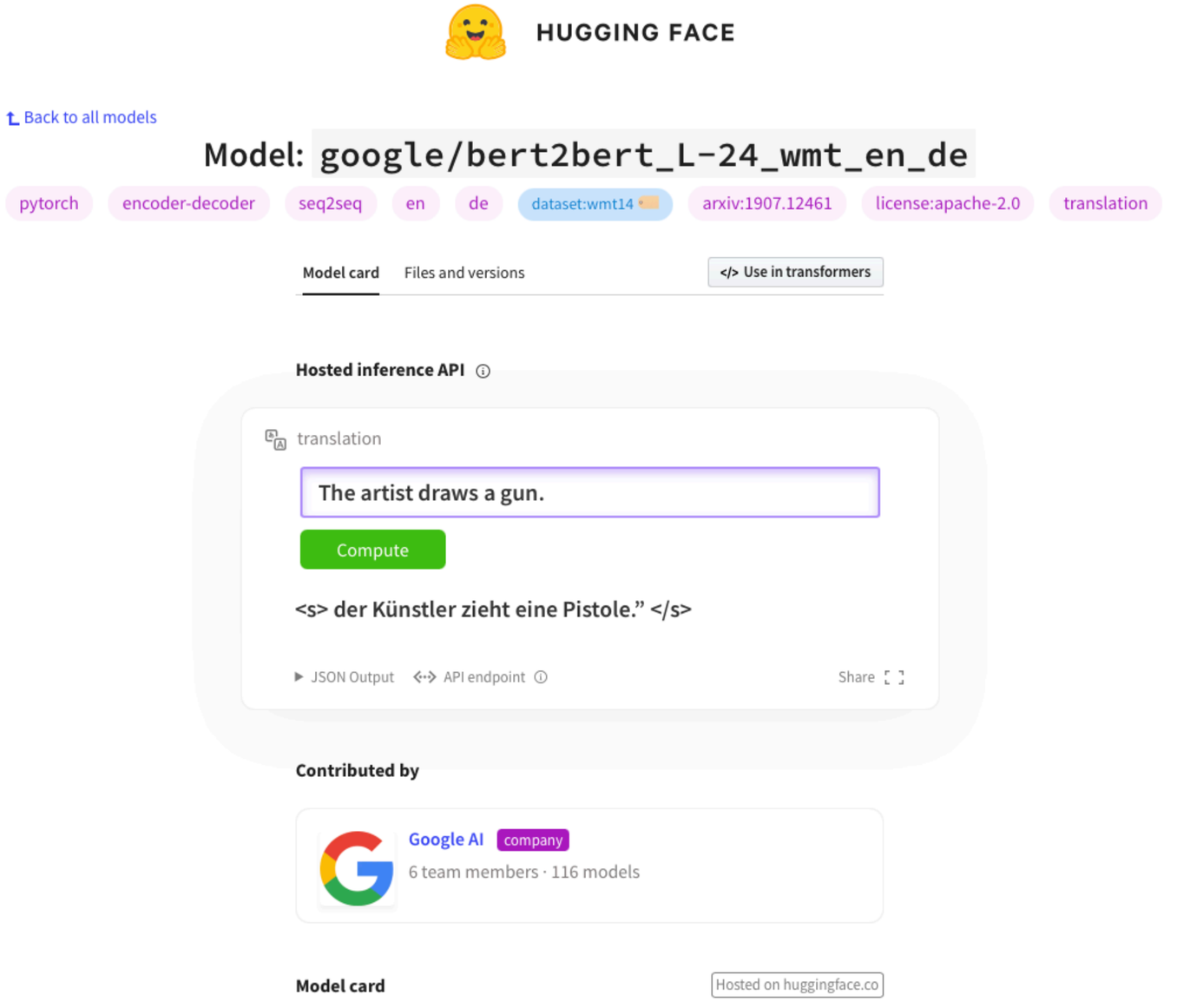
# AMBIGUITY OF WORDS

- The artist draws a gun.
- Two meanings
  - Artistic rendering
  - A dangerous situation

# DEMO TIME: BERT DISAMBIGUATES

**DRAWS -> ZIEHT**

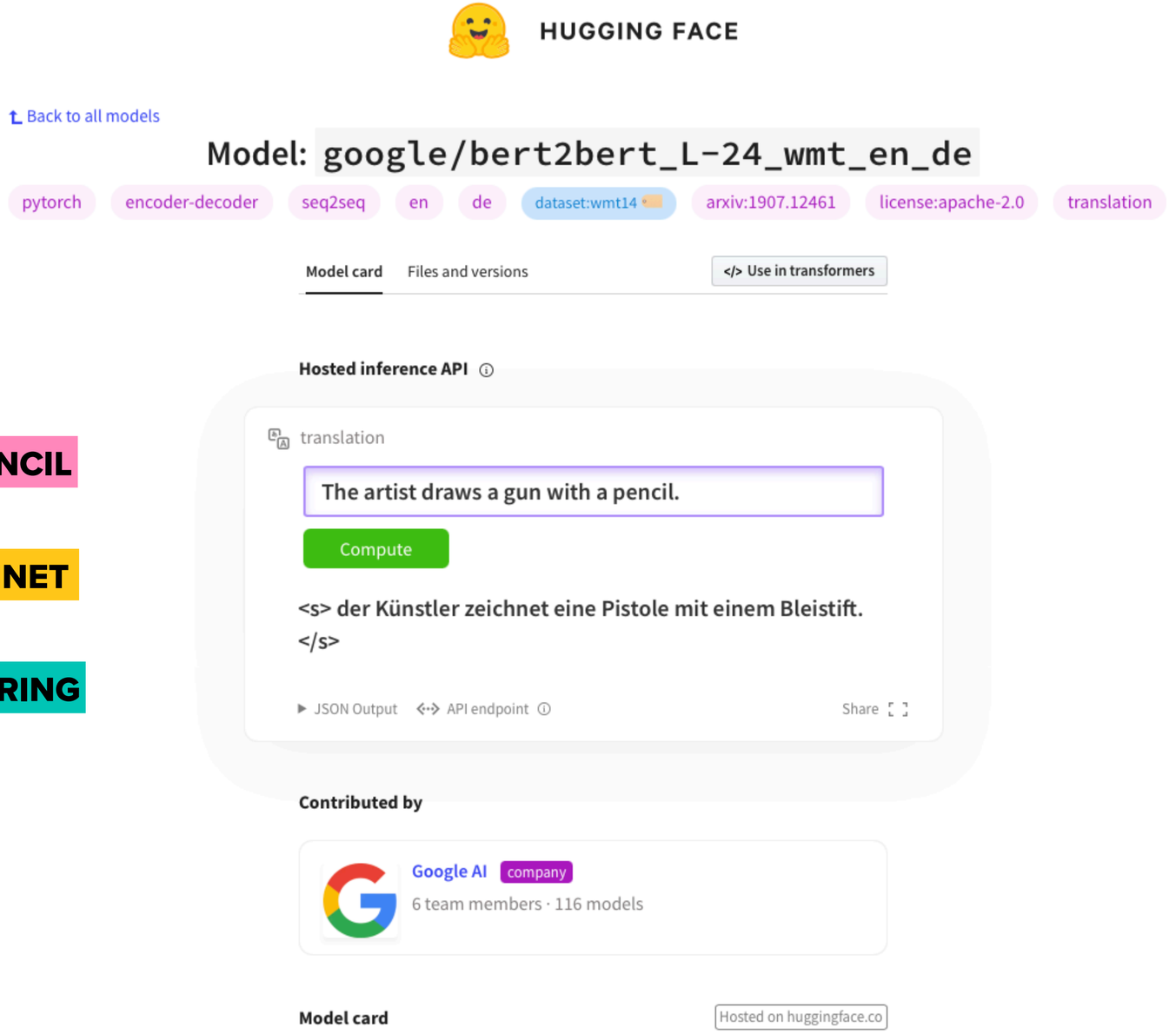
**DANGEROUS SITUATION**



ADD: WITH A PENCIL

DRAWS -> ZEICHNET

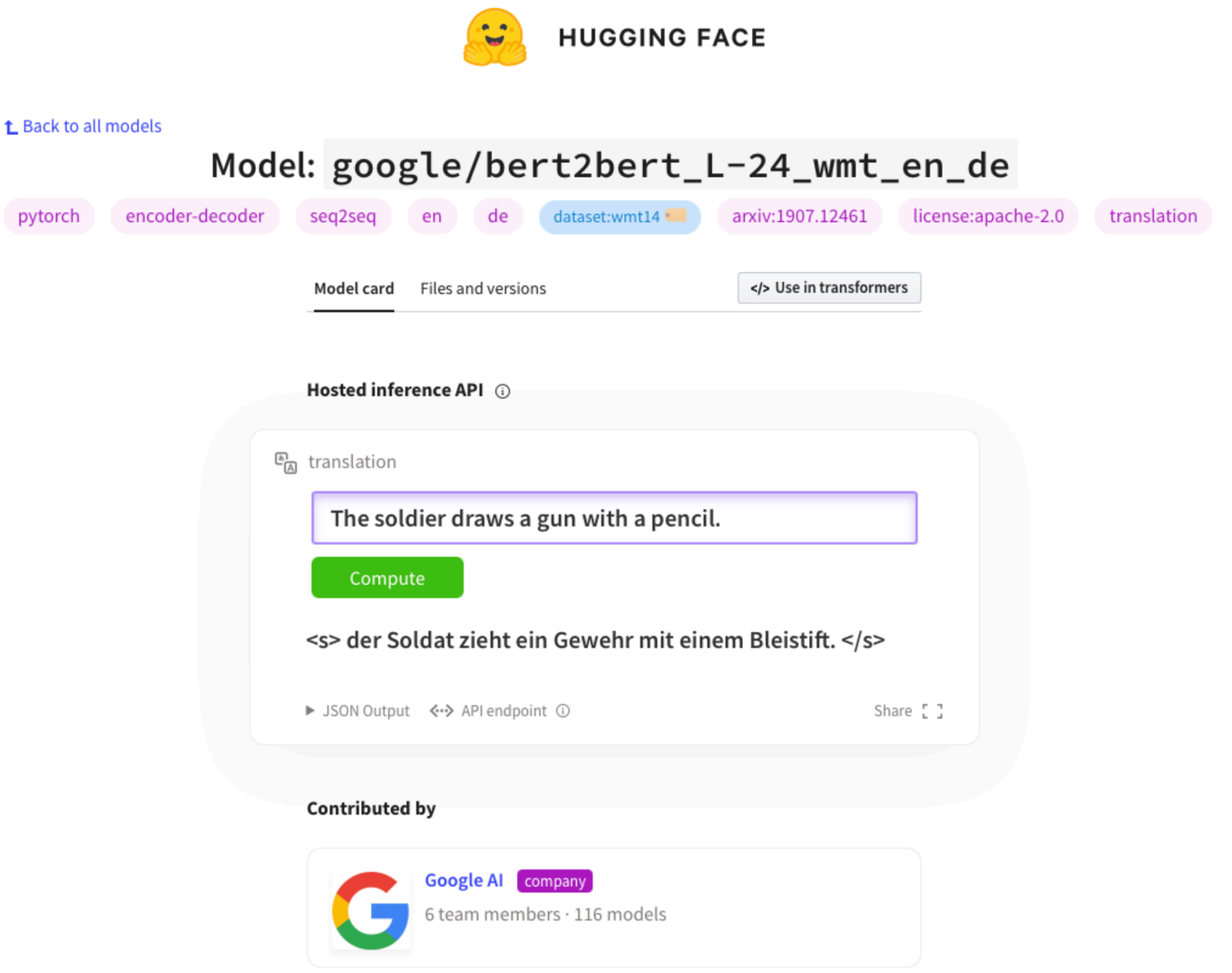
ARTISTIC RENDERING



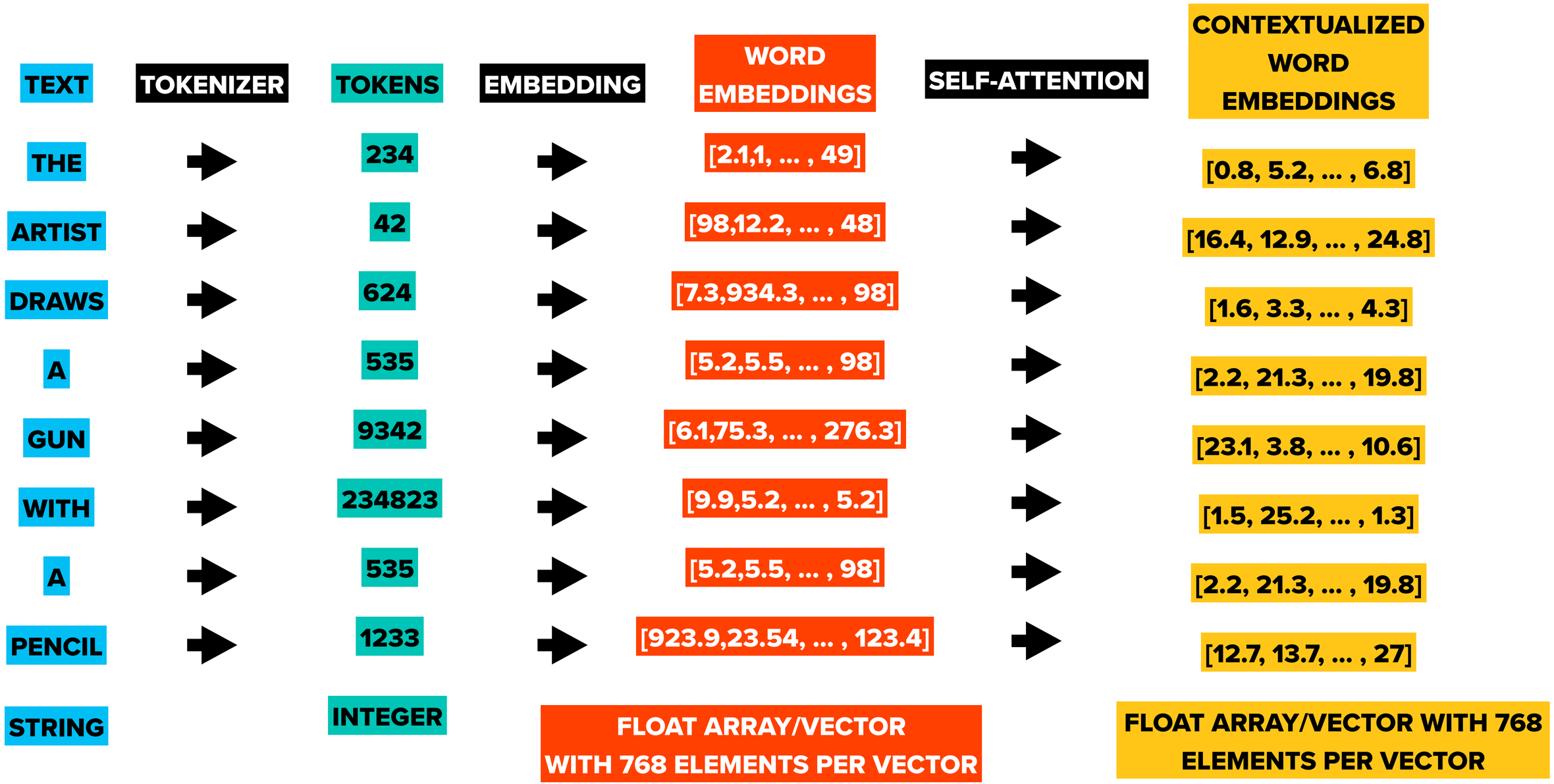
REPLACE: ARTIST
BY: SOLDIER

DRAWS -> ZIEHT

DANGEROUS SITUATION

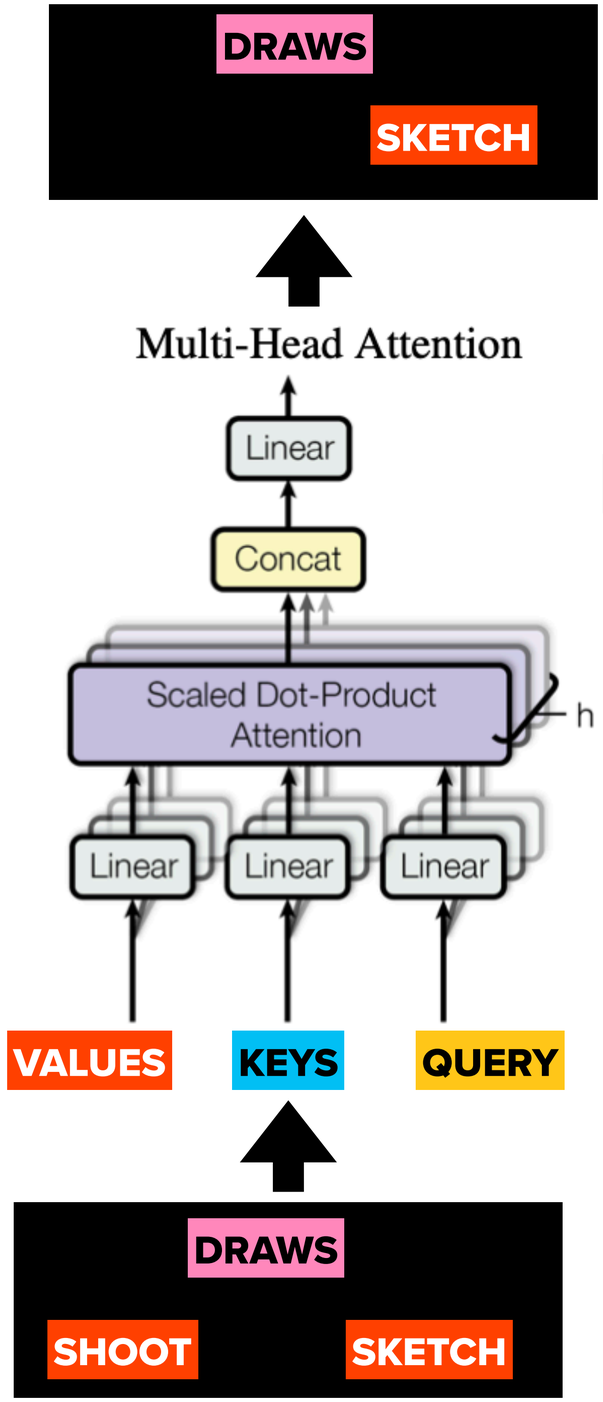


# SELF-ATTENTION I



# Attention Is All You Need

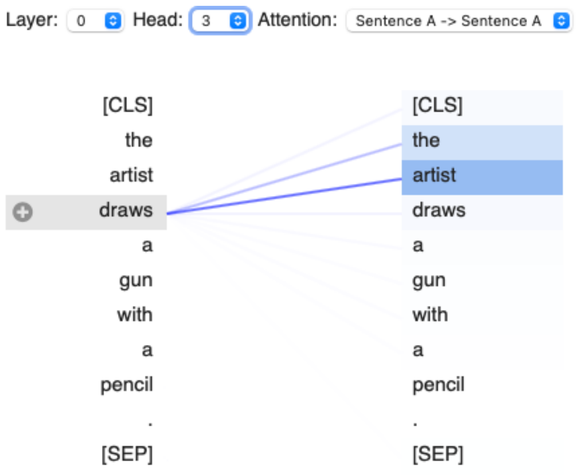
**Ashish Vaswani*** Google Brain avaswani@google.com
**Noam Shazeer*** Google Brain noam@google.com
**Niki Parmar*** Google Research nikip@google.com
**Jakob Uszkoreit*** Google Research usz@google.com
**Llion Jones*** Google Research llion@google.com
**Aidan N. Gomez*** † University of Toronto aidan@cs.toronto.edu
**Łukasz Kaiser*** Google Brain lukaszkaiser@google.com
**Illia Polosukhin*** ‡ illia.polosukhin@gmail.com

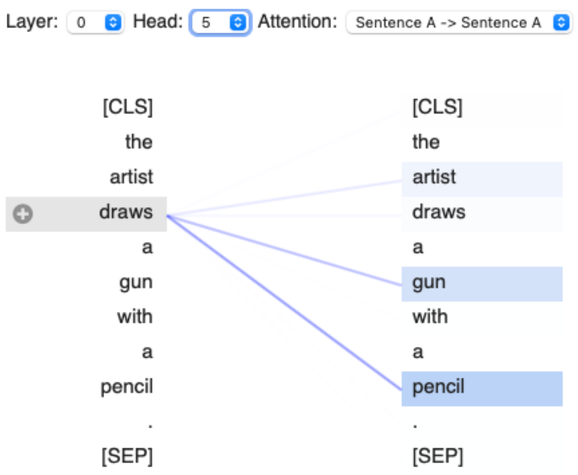


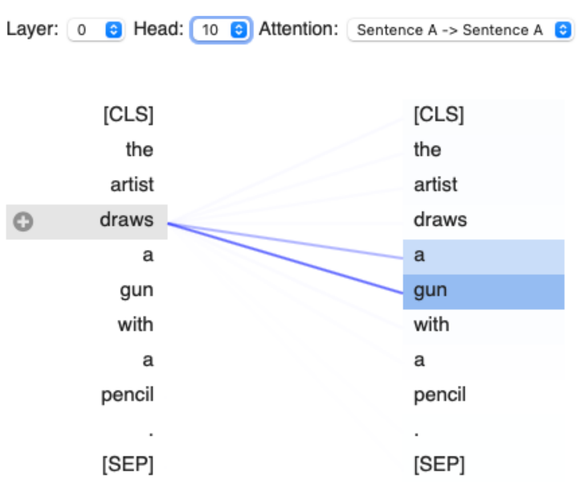

COMBINATION

EACH HEAD QUERIES A DIFFERENT QUESTION AND EXTRACTS A POSSIBLY DIFFERENT MEANING

# SELF ATTENTION - ARTIST




**QUERY**

**KEYS**

**VALUES**

**WHO DRAWS?**

**ARTIST**

**SKETCH**

**HOW IS IT DRAWN?**

**PENCIL**

**SKETCH**

**WHAT IS DRAWN?**

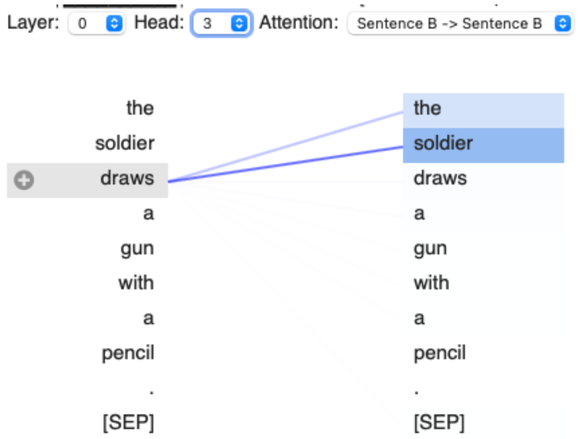
**GUN**

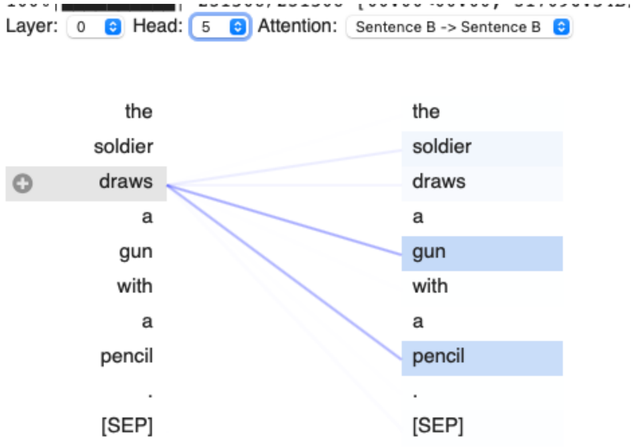
**SHOOT**

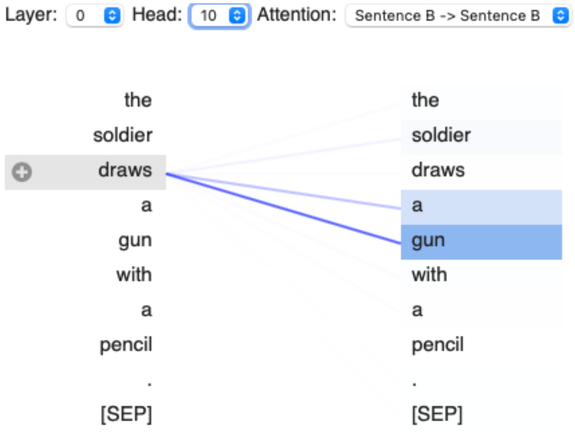
**COMBINATION / LINEAR LAYER**

**SKETCH**

# SELF ATTENTION - SOLDIER




**QUERY** **WHO DRAWS?** **HOW IS IT DRAWN?** **WHAT IS DRAWN?**

**KEYS** **SOLDIER** **PENCIL** **GUN**

**VALUES** **SHOOT** **SKETCH** **SHOOT**

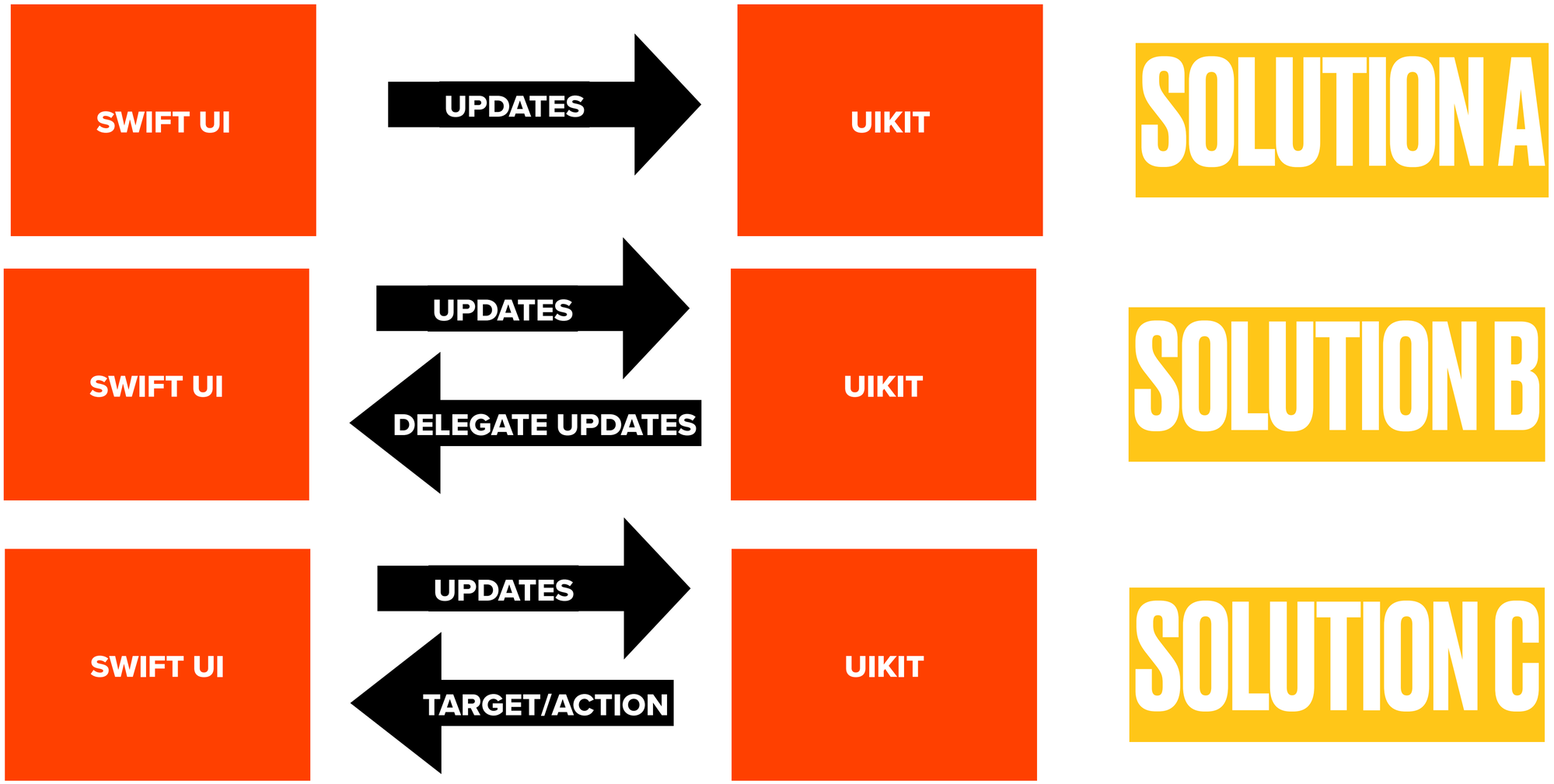
**COMBINATION / LINEAR LAYER**

**SHOOT**

# CODING 1: THE UI

# CODING 2: THE PDF COMPONENT

# HOW TO INTEGRATE A UIKIT COMPONENT IN SWIFT UI



# SOLUTION A

- 1 Create a struct that holds the inputs as attributes with @State and @Binding
- 2 Extend the struct with the protocol
  - UIView: UIViewRepresentable
  - UIViewController: UIViewControllerRepresentable
- 3 Implement
  - makeUIView: return initialized UI View, use inputs for initializations
  - updateUIView: called when attributes change, so that you can react to those changes

# SOLUTION B

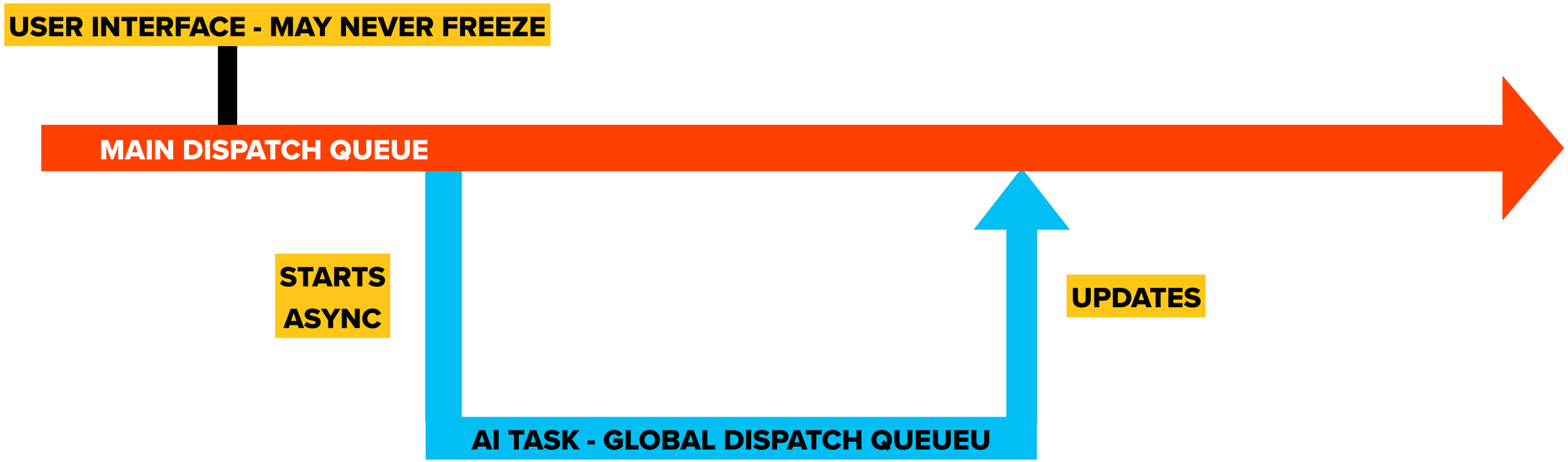
- If you want to provide a delegate to your UIKIt Component
- Implement delegate
  - Add @Binding attributes that you like to update
- Implement makeCoordinator
  - initialize  delegate
  - pass @Binding attributes that you like to update
  - return delegate
- In makeUIViewController/makeUIView
  - register delegate with UIKit Component

# SOLUTION C

- If you need to update your Swift Component via Target Action Pattern
- Implement a new class inheriting from NSObject
  - In it’s init method, pass your UIViewRepresentable/UIViewControllerRepresentable
  - add a @objc function
    - with sender of UIKit Component type and
    - update UIViewRepresentable/UIViewControllerRepresentable based on UIKIt Components changed value
- Implement makeCoordinator
  - initialize Coordinator with self
- In makeUIViewController/makeUIView
  - register your Coordinator @objc function for your update event such as .valueChanged

# CODING 3: USING BERT WITH GCD

# UNDERSTANDING THE GCD, JUST ENOUGH




QUALITY OF SERVICE - QOS - PRIORITY

USER-INTERACTIVE > USER-INITIATED > UTILITY > BACKGROUND

# SINGLE AI TASK PATTERN

```
DispatchQueue.global(qos: .utility).async {
    // perform AI task
    DispatchQueue.main.async {
        //update UI
    }
}
```

# UNDERSTANDING CONCURRENCY

- Global vs Private Dispatch Queues
  - Level of Concurrency: serial vs concurrent
  - Number of concurrent tasks
- Serial vs Concurrent Queues
  - Serial: one task at a time
  - Concurrent: multiple tasks at a time
- Sync vs Async
  - Sync: blocks the caller queue until task is finished (never on Main Queue)
  - Async: returns immediately, no blocking

# DISPATCH GROUPS

- Run multiple AI tasks asynchronously
- Update UI when all of them are finished
- Use
  - DispatchGroup() to create group
  - for each task, group.enter() and group.leave()
  - make sure that you always leave the task by using defer
  - enter and leave serve as a concurrent counter for the group to know when all tasks are finished
  - use group.notify() to respond to the event that all tasks are done

```swift
import UIKit

let aiTasks = DispatchGroup()
var results = [String]()

let inputs = ["Text 1", "Text 2", "Text 3" ]
let question = "Is the number of multiverses finite?"


for input in inputs {
    aiTasks.enter()
    DispatchQueue.global(qos: .utility).async {
        defer { aiTasks.leave() }
        // do your ai task
        sleep(1)
        print("finished: \(input)")
        results.append("\(input) is the answer to \(question)")
    }
}
print("we are async")
aiTasks.notify(queue: DispatchQueue.main) {
    // update ui
    print("all done")
    print(results)
}
```

```
we are async
finished: Text 2
finished: Text 3
finished: Text 1
all done
["Text 2 is the answer to Is the number of multiverses finite?", "Text 3 is the answer to Is the number of multiverses finite?", "Text
1 is the answer to Is the number of multiverses finite?"]
```

# INTERMISSION

# WHERE TO GO FROM HERE

# THE COFFEE APP

# COFFEE APP PURPOSE

- Pour Over Coffee - think of third wave coffee shops such as Blue Bottle
- At home you need to brew it with fairly precise measurements
- Each coffee may have a perfect sweet spot for each measure so you need to experiment a lot
- We will look at
  - Temperature
  - Grind size
- And build an app that functions as a diary for those measurements and star it based on coffee perfection



# RESNET

# PART 3: THE COFFEE APP

App Specifications

ResNet

Coding 1: The UI

Coding 2: Saving your Data - Designed for Privacy

Coding 3: Real-Time Camera Access

Coding 4: Integrating Vision Framework

# Deep Residual Learning for Image Recognition

Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun
Microsoft Research
{kahe, v-xiangz, v-shren, jiansun}@microsoft.com

Deep residual learning for image recognition
K He, X Zhang, S Ren, J Sun - Proceedings of the IEEE ..., 2016 - openaccess.thecvf.com
Deeper neural networks are more difficult to train. We present a residual learning framework to ease the training of networks that are substantially deeper than those used previously. We explicitly reformulate the layers as learning residual functions with reference to the layer inputs, instead of learning unreferenced functions. We provide comprehensive empirical evidence showing that these residual networks are easier to optimize, and can gain accuracy from considerably increased depth. On the ImageNet dataset we evaluate residual ...
☆ Cited by 64880 Related articles All 68 versions


He, Kaiming, et al. "Deep residual learning for image recognition." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.

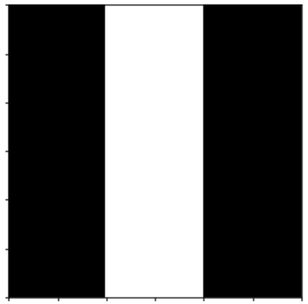
# RESNET

- Background Knowledge:
  - CNN - Convolutional Neural Networks
  - CNNs learn filters with hierarchical learning
  - CNNs are are Deep Neural Network
  - Deep Neural Networks are Neural Networks with many layers
- Problem:
  - Deep Neural Networks can suffer from Vanishing Gradient Problem
  - This causes the network not to learn any filters
- Solution:
  - ResNet addresses vanishing gradient problem with
  - Residual / Skip - Connections

# WHAT DOES A FILTER DO?

+

3X3 FILTER

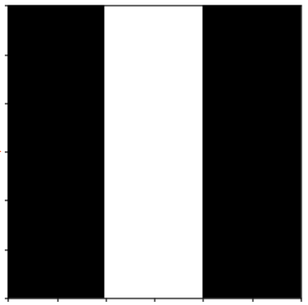


CONVOLUTION

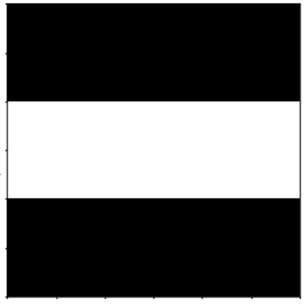
INPUT + FILTER FEATURE CONVOLUTION ACTIVATION MAP

# DIFFERENT FILTERS IN ACTION

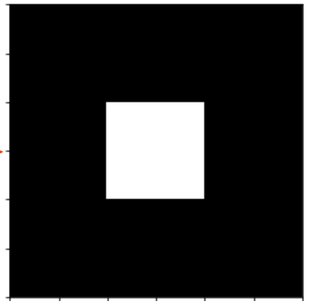
HORIZONTAL

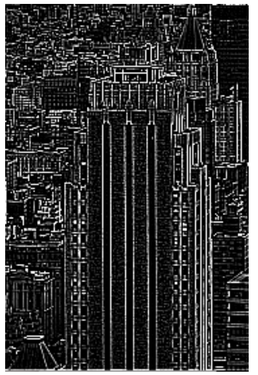


VERTICAL

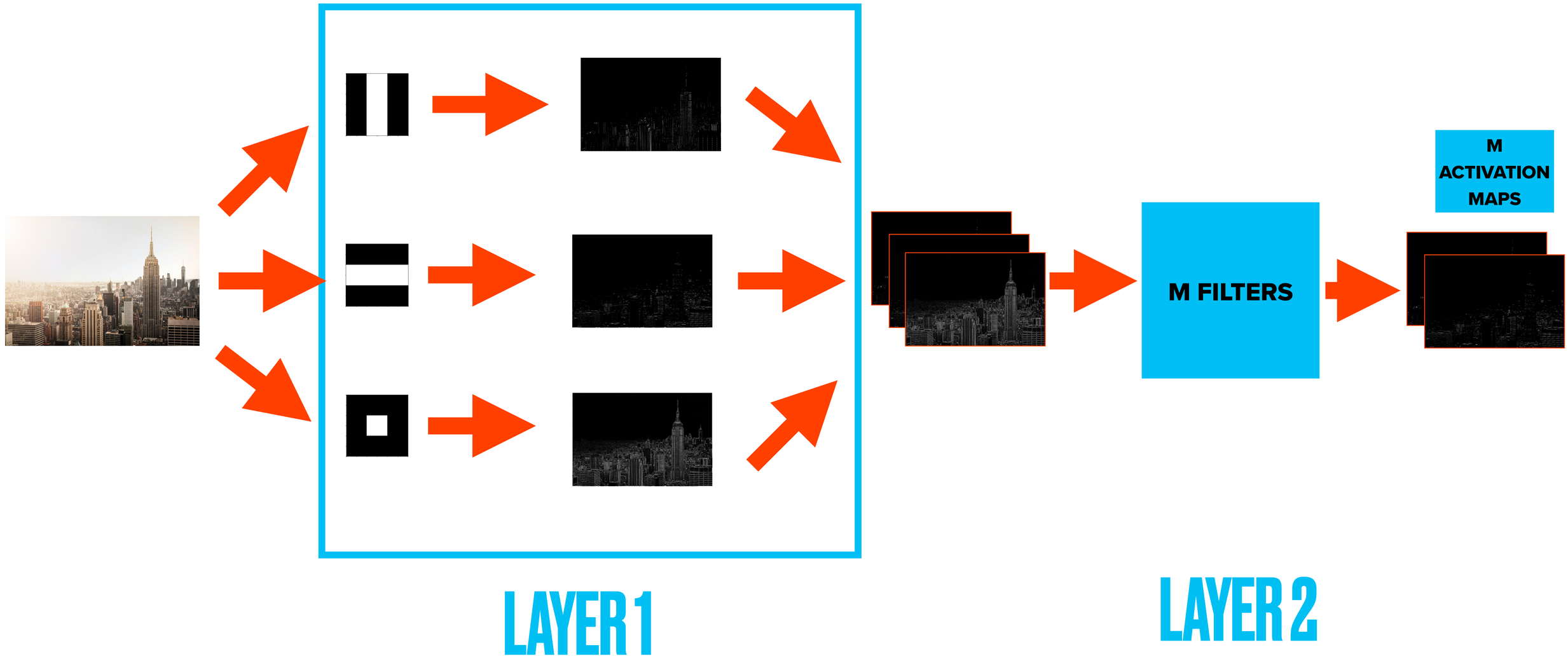


EDGES





# HIERARCHICAL LEARNING



# HIERARCHICAL LEARNING - THE HOW


The more abstract the problem, the more layers we will need

# HIERACHICAL LEARNING - THE WHY

## UNIVERSAL APPROXIMATION THEOREM

HIERACHICAL NEURAL NETWORK

SINGLE LAYER NETWOR

Reasons for Hierachical Networks

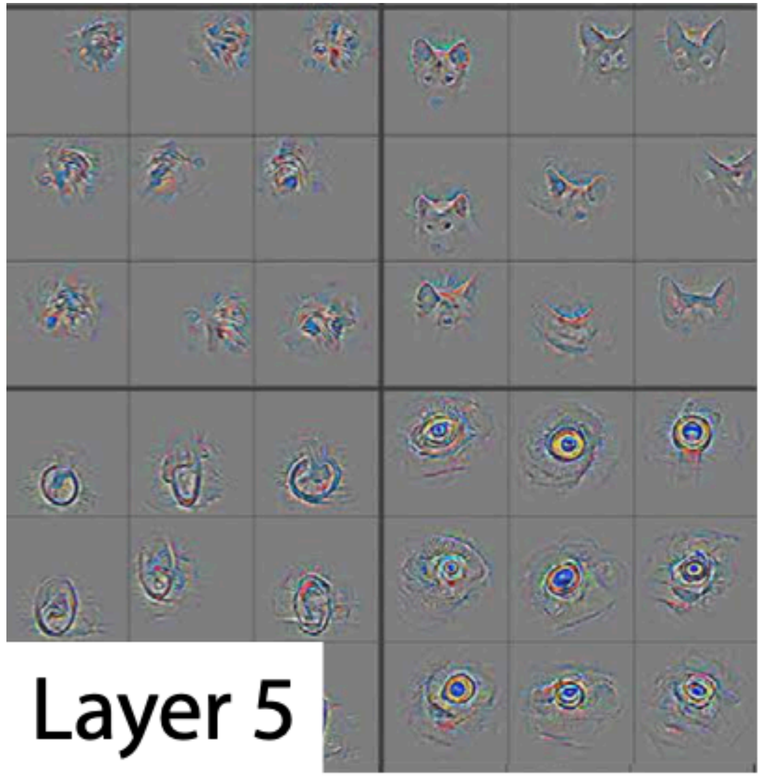
1. Globally: Fewer Neurons / Parameters
2. Locally: Fewer Neurons per Layer - better use of GPUs

IN THE PAST WE CREATED FILTERS MANUALLY

CNNS LEARN FILTERS FROM DATA

# WHAT FILTERS DO CNNS LEARN?

Layer 1



Layer 5

**Visualizing and Understanding Convolutional Networks**

**Matthew D. Zeiler** ZEILER@CS.NYU.EDU
Dept. of Computer Science, Courant Institute, New York University

**Rob Fergus** FERGUS@CS.NYU.EDU
Dept. of Computer Science, Courant Institute, New York University

# VANISHING GRADIENT PROBLEM

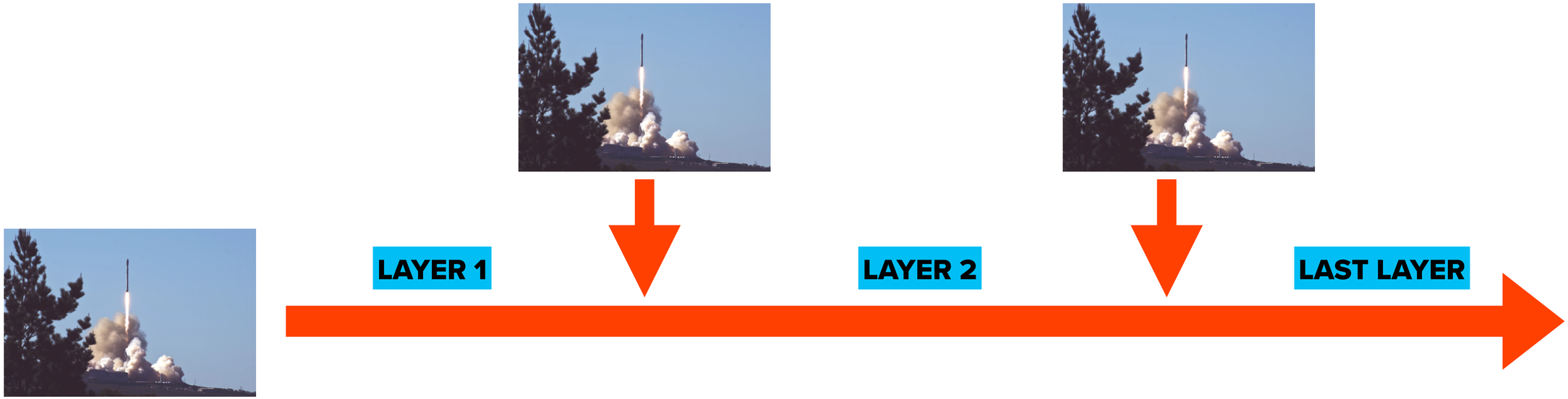
ALL INFORMATION FILTERED FEATURES

IMAGE HAS CHROME COLOR

CHROME IS NOT A FEATURE

IS IT A FALCON 9 ROCKET?

# RESIDUAL / SKIP CONNECTIONS




1. Addresses the vanishing gradient problem due to proportional update of backpropogation
2. Transfer Learning: if you use a network not trained for your task, skip connection allow quicker convergence in practice because they can use the chrome just in the last layer and train only this layer anew instead of changing all previous layers

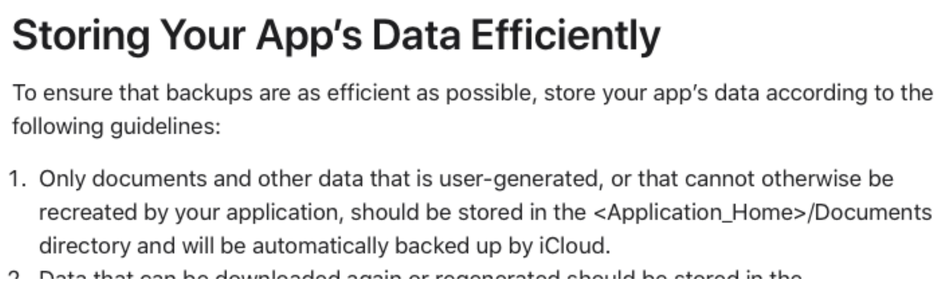
# CODING 1: THE UI

# DESIGN FOR “APPLE GLASSES”

# CODING 2: SAVING YOUR DATA DESIGNED FOR PRIVACY

# WHERE TO STORE USER IMAGES

- iOS Data Storage Guidelines



### Storing Your App's Data Efficiently

To ensure that backups are as efficient as possible, store your app's data according to the following guidelines:

1. Only documents and other data that is user-generated, or that cannot otherwise be recreated by your application, should be stored in the <Application_Home>/Documents directory and will be automatically backed up by iCloud.
2. Data that can be downloaded again or regenerated should be stored in the

- The images are user-generated
- Saving them in the documents directory allows backup in iCloud
- 

```
FileManager.default.urls(for: .documentDirectory, in: .userDomainMask).first
```

file:///Users/stefan/Library/Developer/XCPGDevices/88E

# JSON SERIALIZATION

- 1: Adopt the Codable protocol on the struct/class you want to save or load from disk
- 2: Do you have any custom types as attributes?
  - Then those custom types also need to adopt the Codable protocol
  - Native types already support it
- To save to disk:

```
let encoder = JSONEncoder()
do {
    let beansData = try encoder.encode(coffeeBeans)
    try beansData.write(to: collectionUrl, options: .atomicWrite)
} catch let error {
    // here we need to add proper error handling in production applications
    print(error)
}
```

- To load from disk:

```
let decoder = JSONDecoder()

do {
    let beansData = try Data(contentsOf: collectionUrl)
    coffeeBeans = try decoder.decode([CoffeeBean].self, from: beansData)
} catch let error {
    // here we need to add proper error handling in production applications
    print(error)
}
```

# CODING 3: REAL-TIME CAMERA ACCESS

# CAMERA RECORDING SESSION/ CAMERA CONTROLLER

- Inherit from AVCaptureVideoDataOutputSampleBufferDelegate
- Setup an AVCaptureSession, add an input device and configure them as needed
- Queue AVCaptureVideoDataOutput to your VNImageRequestHandler

```
let session = AVCaptureSession()
private let videoDataOutput = AVCaptureVideoDataOutput()
let videoDataOutputQueue = DispatchQueue(label: "VideoDataOutput", qos: .userInitiated, attributes:
[], autoreleaseFrequency: .workItem)
let videoDevice = AVCaptureDevice.DiscoverySession(deviceTypes: [.builtInWideAngleCamera],
mediaType: .video, position: .back).devices.first

videoDataOutput.setSampleBufferDelegate(self, queue: videoDataOutputQueue)

session.addInput(deviceInput)
let captureConnection = videoDataOutput.connection(with: .video)


let requestHandler = VNImageRequestHandler(cgImage: cgImage,  options: [:])
let request = VNGenerateImageFeaturePrintRequest()
do {
    try requestHandler.perform([request])
    return request.results?.first as? VNFeaturePrintObservation
}
```

# CODING 4: INTEGRATING VISION FRAMEWORK

# VISION FRAMEWORK REQUESTS

- Follows the same GCD pattern from the legal app
- requires a CGImage
- Image types
  - UIImage: High Level Image for representation in UI (png/jpeg)
  - CGImage: Quartz CoreGraphic images, needs to be a bitmap
  - CIImage: Core Image which is basically data that can be transformed into an image

```swift
import UIKit
import Vision

let image = (UIImage(named: "image1.jpeg")?.cgImage)!

let imageRequestHandler = VNImageRequestHandler(cgImage: image, orientation: .right, options: [:])
let request = VNRecognizeTextRequest { req, error in
    guard let observations = req.results as? [VNRecognizedTextObservation] else {
        return
    }
    let possibleResults: [Int] = observations.compactMap { observation in
        var result: Int? = nil
        for candidate in observation.topCandidates(15) {
            guard let number = Int(candidate.string) else {
                continue
            }
            result = number
            break
        }

       return result

    }
    if possibleResults.count < 2 {
        return
    }
    let current = possibleResults.min()!
    let target = possibleResults.max()!

    DispatchQueue.main.async {
        print("Current temperature: \(current)")
        print("Target temperature: \(target)")

    }
}
do {
    try imageRequestHandler.perform([request])
} catch {
    print("error occured during processing")
}
```

# WHERE TO GO FROM HERE

# PART 4: DESIGN FOR PRIVACY

Approaches

What we didn’t do in our Apps

Neural Engine

Importance of Privacy Now

Future-Proof Privacy

# APPROACHES TO USING AI ON MOBILE DEVICES

- CLOUD APPROACH
- ON-DEVICE

# WHAT WE DIDN'T DO IN OUR APPS

- We didn't store the images in our cloud
- We didn't process the images in our via REST service for our AI in the cloud
- Pros
  - we don't pay for storage for the images
  - we don't have costs for the API calls and CPU/GPU resources
  - we don't have service outages or maintenance costs
- Cons
  - We cannot improve our AIs based on the images from users
  - It's harder to argue for subscription costs based on data storage & api calls
  - Some AI tasks may be too complex for the phone's neural engine to handle

# NEURAL ENGINE

- Using the AIs on device is only possible with a specialized hardware that can run AIs efficiently
- The Power of The Neural Engine
- Ensures Privacy and Reduces our deployment costs
- However our AIs, need to be adjusted so that they are best compatible with the Neural Engine
- We need to make sure that we don’t drain the battery and still have a responsive app

# THE IMPORTANTCE OF PRIVACY - NOW

- Self-fulfilling prophecy: Some people actually care about privacy (even in the 🇺🇸)
  - These users might even be skewed towards Apple's ecosystem - target them!

- Designing for privacy means NOT HAVING TO TAKE responsibility for
  - Data Leaks / Hacker attacks
  - Data Management / Big Data (do you want to end up with Hadoop?)
  - Sending stuff back and forth
  - Your users' privacy!

- By valuing privacy now, you build trust for your brand later!

# THE IMPORTANCE OF PRIVACY IN THE FUTURE

- Let’s imagine Apple comes up with something you wear on / in your eyes
  - It is permanently recoding - everything
  - Eventually, nobody would want to have everything they see shared with profit seeking companies
  - Apple might not allow it, laws might prohibit it!
- There might be a future with body parts that constantly process data which you most definitely don’t want to share
  - In-Body Device DNA/RNA Testing

- The future does not have to be a dystopia
- Privacy preserving intelligence might become mandatory.