**#Introduction**

I built a dog vs. cat classification system using transfer learning. Transfer learning is a very important concept in deep learning where we use pre-trained models in order to build our use cases. So in this case, I used a pre-trained model called **MobileNet V2**, where we will train that model on images of dogs and cats. In turn, this model can tell you whether an image represents a dog or a cat. So this is what we are going to discuss in this project. So first of all, let's try to understand what this transfer learning is, and later I'll explain to you what is the workflow that we followed for this use case. Later on, we can move on to the hands-on part where we will build this system using Python. So first of all, let's try to understand what this transfer learning is.

# #Transfer Learning

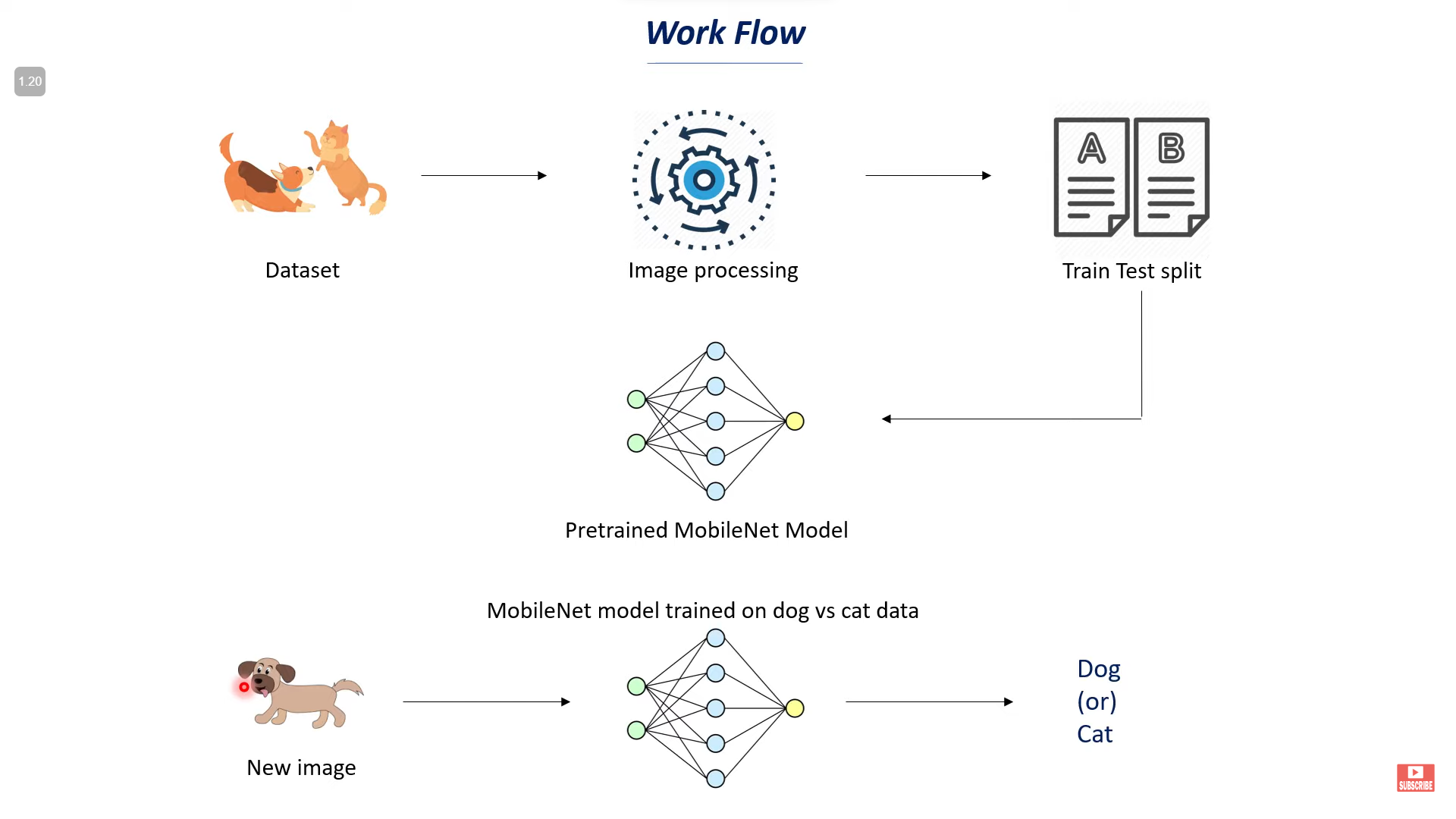
Transfer Learning is a Deep Learning technique where we use a pre-trained model.

This pre-trained model is trained for one task and can be re-trained for a similar task

with a smaller dataset.

Transfer Learning gives higher accuracy compared to training models from scratch.

# #Workflow



As shown in the slide, we start by collecting pictures of dogs and cats. We're using the Kaggle API to download these pictures from a website called Kaggle.

Once we have all these pictures, the next big step is to get them ready for the computer. We change their size so they all match and turn them into a type of data the computer can use.

Next, we split these pictures into two bunches. One bunch is for teaching our computer thing, and the other bunch is to check if it's learning well. The teaching bunch has pictures labeled as either 'dog' or 'cat.'

Then, we use a ready-made computer thing called MobileNet V2. It's good for this job because it works fast and well. We make it even smarter by teaching it with our dog and cat pictures.

After all the teaching, we check how good it is with the checking bunch of pictures. We want to see if it can get it right with new pictures it hasn't seen before.

In the end, our computer thing should be smart enough to look at a new picture and tell if there's a dog or a cat in it.

# #Steps of Code

1. **Getting Pictures**: First, I used a tool called Kaggle API to download lots of dog and cat pictures from the internet.
2. **Preparing the Pictures**: After downloading, I organized the pictures and made them all the same size so that our system can learn from them easily.
3. **Looking at the Pictures**: I made sure that we could see the pictures on the computer and check if they are dogs or cats.
4. **Counting Dogs and Cats**: I wrote some code to count how many dog pictures and how many cat pictures we have.
5. **Making Pictures Ready for Learning**: I changed all the pictures into a special format that the computer likes, called numpy arrays.
6. **Splitting the Pictures**: I split the pictures into two groups: one for teaching the computer and one for testing if the computer has learned well.
7. **Teaching the Computer**: I used a smart model called MobileNet that's already learned about many things. I told it to focus on learning about dogs and cats.
8. **Checking the Learning**: I checked how good the computer got at telling dogs from cats with the test pictures.
9. **Making Guesses**: Finally, I made a part where you can give the computer a new picture, and it'll guess if it's a dog or a cat.

And that's it! That's how I taught the computer to tell the difference between dogs and cats.

# #Steps of code (deep)

1. **Setting Up the Environment**: To start, I installed necessary libraries like kaggle to access the image datasets.
2. **Accessing the Data**: I set up a secure connection to Kaggle using a special file (kaggle.json) which lets me download the dataset directly into our project environment.
3. **Downloading and Preparing the Dataset**: I used the Kaggle command-line tool to download the 'Dogs vs Cats' dataset. Then, I extracted the images from a zip file, so they are ready to be processed.
4. **Exploring the Dataset**: I looked inside the dataset to see what pictures we have and counted how many there are using Python's os library.
5. **Viewing Sample Images**: To understand what the images look like, I displayed a few of them directly in the notebook.
6. **Preprocessing**: I resized all the images to a uniform size (224x224 pixels), which is important because the computer needs them to be the same size to learn properly.
7. **Labeling the Images**: I assigned labels to the images – '0' for cats and '1' for dogs – because the computer likes numbers more than words.
8. **Creating Training and Testing Sets**: I split the labeled images into two sets: one set for teaching the computer (training) and one set to see how well it learned (testing).
9. **Normalizing the Data**: Before teaching the computer, I adjusted the images' values so they're easy for the computer to work with (a process called scaling).
10. **Building a Brain for Our Computer (Neural Network):** I then used a famous and smart model called MobileNet V2. It's like a brain that's already learned a lot, and now it will learn more about dogs and cats.
11. **Training the Model**: I taught our computer brain with our training set of dog and cat images.
12. **Evaluating the Model**: After teaching, I tested the computer brain with our test set to see how well it could tell the difference between new pictures of dogs and cats.
13. **Predicting New Images**: Lastly, I made a part where you can give the computer a picture it's never seen before, and it will use what it learned to tell if it's a dog or a cat.