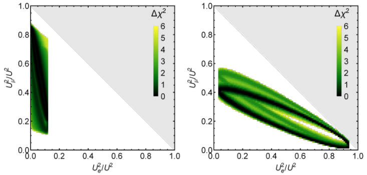
Title

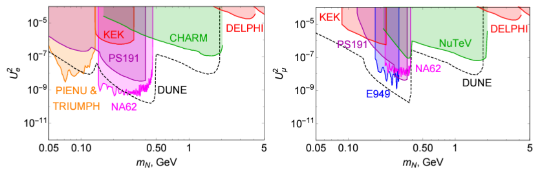
Alex Mikulenko

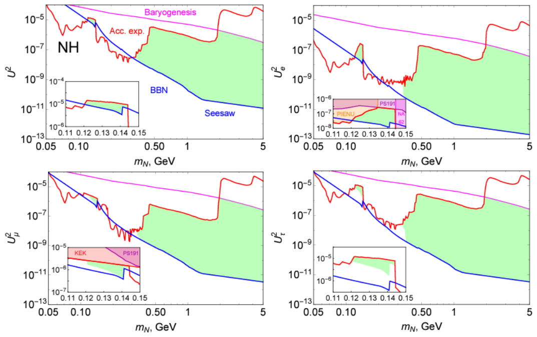
Supervisor:
Alexey Boyarsky

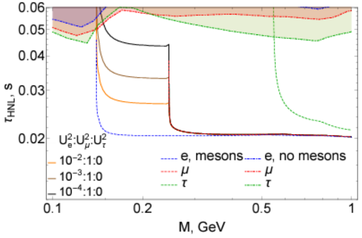
05.2023

[2101.09255]
An allowed window for heavy neutral leptons below the kaon mass, Bondarenko et al.



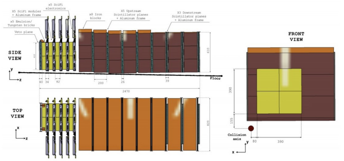


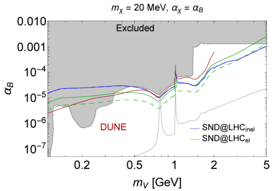


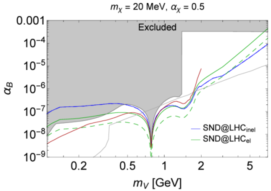


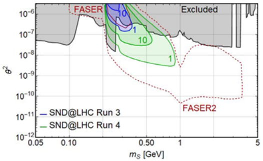
[2104.09688]

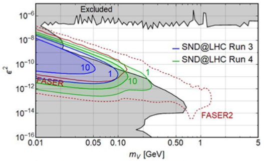
Searches for new physics at SND@LHC, Boyarsky et al.

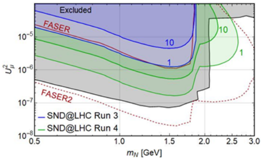


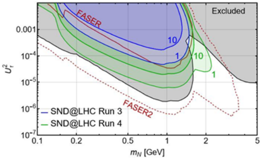






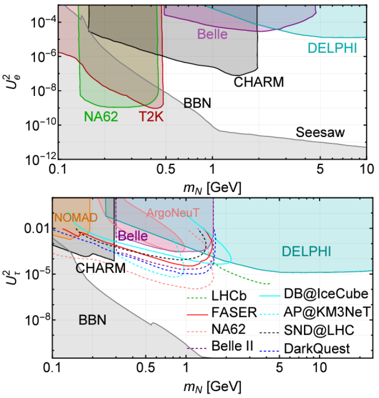




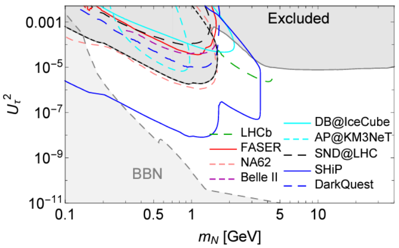


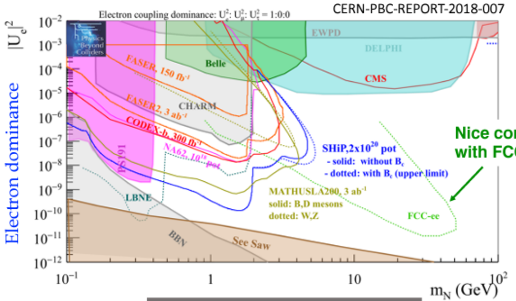
[2107.14685]

Constraints from the CHARM experiment on Heavy Neutral Leptons with tau mixing, Boiarska et al.



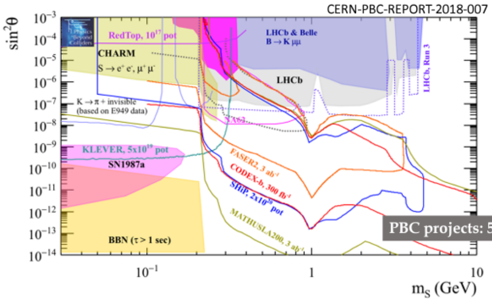
$\Longrightarrow$





Nice complementarit
with FCC-ee

PBC projects: ~10-15 year outlook



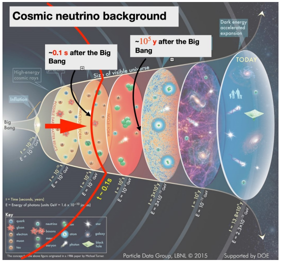
PBC projects: 5-15 years outlook

Other particle physics papers

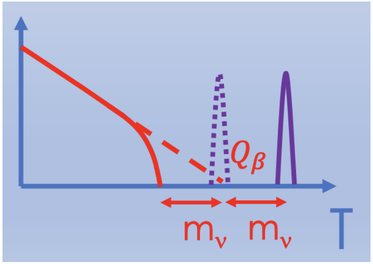
1. The SHiP experiment at the proposed CERN SPS Beam Dump Facility [2112.01487]
2. The Forward Physics Facility at the High-Luminosity LHC [2203.05090]
3. Searches for long-lived particles at the future FCC-ee [2203.05502]
4. The present and future status of heavy neutral leptons [2203.08039]
5. Exploring the potential of FCC-hh to search for particles from B mesons [2204.01622]
6. Report of the Topical Group on Physics Beyond the Standard Model at Energy Frontier for Snowmass 2021 [2209.13128]
7. SND@LHC: The Scattering and Neutrino Detector at the LHC [2210.02784]
8. Towards the optimal beam dump experiment to search for feebly interacting particles [2304.02511]

[2111.09292]

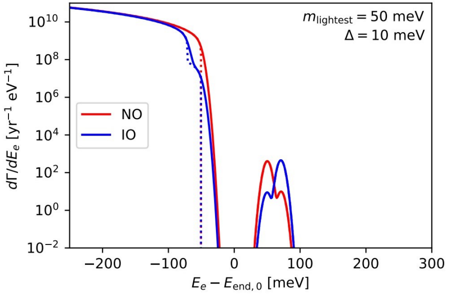
Can we use heavy nuclei to detect relic neutrinos? Mikulenko et al.

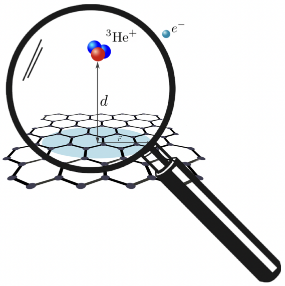


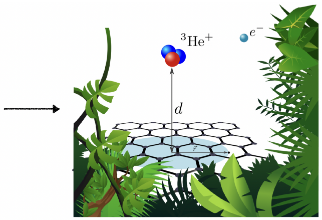
Cosmic neutrino background
~0.1 s after the Big Bang
~10^5 y after the Big Bang
Dark energy accelerated expansion
TODAY
Size of visible universe
High-energy cosmic rays
Inflation
Big Bang
t ~ 0.1s
Particle Data Group, LBNL © 2015
Supported by DOE

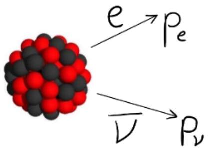


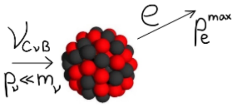
P on-
T ecorvo
O bservatory for
L ight,
E arly-universe,
M assive-neutrino
Y ield





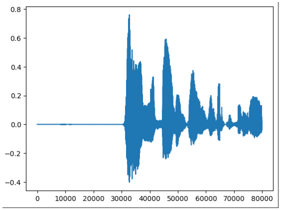




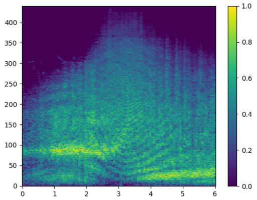


$$\frac{d\Gamma}{dE_e} = \frac{p_\nu E_\nu p_e E_e}{2\pi^3} \times \sum |\mathcal{M}_\mathcal{H}|^2$$

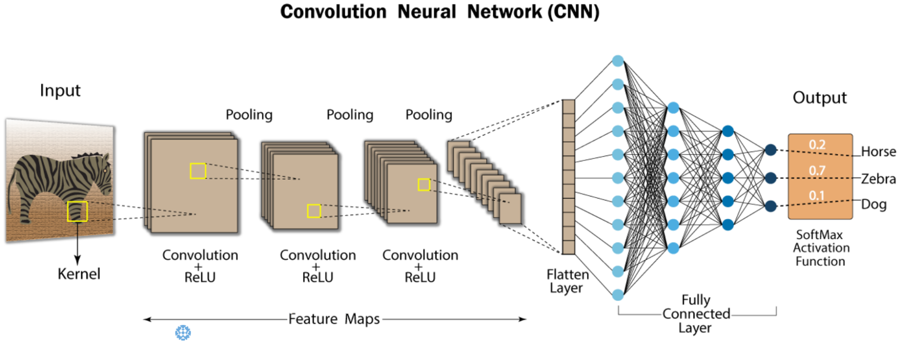
$$(\sigma v)_\nu = \frac{p_e^{\max} E_e^{\max}}{\pi} \times \frac{1}{2} \sum |\mathcal{M}_\mathcal{H}|^2$$



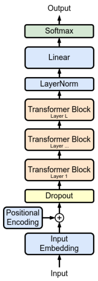
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
0
10000
20000
30000
40000
50000
60000
70000
80000

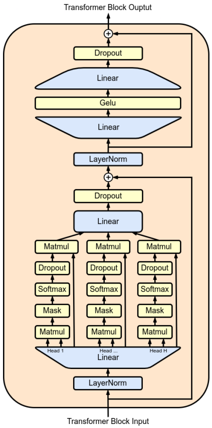


400
350
300
250
200
150
100
50
0
0
1
2
3
4
5
6
1.0
0.8
0.6
0.4
0.2
0.0

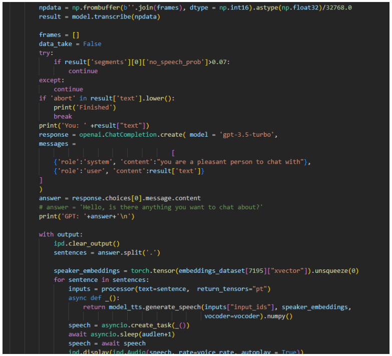


Convolution Neural Network (CNN)
Input
Kernel
Pooling
Pooling
Pooling
Convolution + ReLU
Convolution + ReLU
Convolution + ReLU
Feature Maps
Flatten Layer
Fully Connected Layer
Output
0.2
0.7
0.1
Horse
Zebra
Dog
SoftMax Activation Function



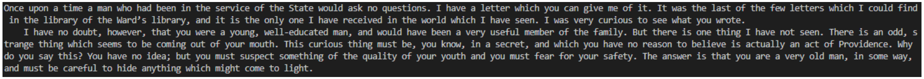


Whisper + GPT4 + audio↔ text = talking GPT4

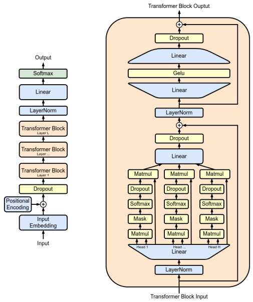


```
npdata = np.frombuffer(b''.join(frames), dtype = np.int16).astype(np.float32)/32768.0
result = model.transcribe(npdata)

frames = []
data_take = False
try:
    if result['segments'][0]['no_speech_prob']>0.07:
        continue
except:
    continue
if 'abort' in result['text'].lower():
    print('Finished')
    break
print('You: ' +result["text"])
response = openai.ChatCompletion.create( model = 'gpt-3.5-turbo',
messages =
                                    [
    {'role':'system', 'content':"you are a pleasant person to chat with"},
    {'role':'user', 'content':result['text']}
]
)
answer = response.choices[0].message.content
# answer = 'Hello, is there anything you want to chat about?'
print('GPT: '+answer+'\n')

with output:
    ipd.clear_output()
    sentences = answer.split('.')

    speaker_embeddings = torch.tensor(embeddings_dataset[7195]["xvector"]).unsqueeze(0)
    for sentence in sentences:
        inputs = processor(text=sentence,  return_tensors="pt")
        async def _():
            return model_tts.generate_speech(inputs["input_ids"], speaker_embeddings,
                                            vocoder=vocoder).numpy()
        speech = asyncio.create_task(_())
        await asyncio.sleep(audlen+1)
        speech = await speech
        ipd.display(ipd.Audio(speech, rate=voice_rate, autoplay = True))
```

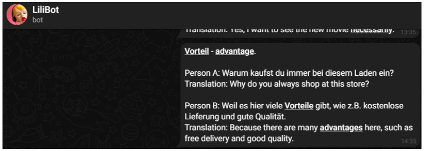
Retrain GPT2 or train your own GPT from scratch



```
Once upon a time a man who had been in the service of the State would ask no questions. I have a letter which you can give me of it. It was the last of the few letters which I could find
 in the library of the Ward's library, and it is the only one I have received in the world which I have seen. I was very curious to see what you wrote.
    I have no doubt, however, that you were a young, well-educated man, and would have been a very useful member of the family. But there is one thing I have not seen. There is an odd, s
trange thing which seems to be coming out of your mouth. This curious thing must be, you know, in a secret, and which you have no reason to believe is actually an act of Providence. Why
do you say this? You have no idea; but you must suspect something of the quality of your youth and you must fear for your safety. The answer is that you are a very old man, in some way,
and must be careful to hide anything which might come to light.
```



Or make a bot that sends your generated human-like examples of usage of words that your want to learn using API to GPT-3.5



LiliBot
bot

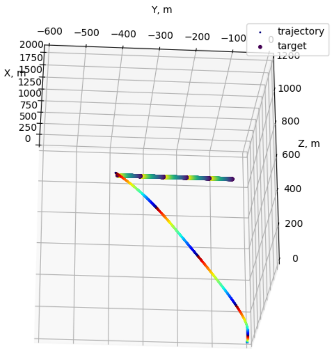
Translation: Yes, I want to see the new movie necessarily. 13:35

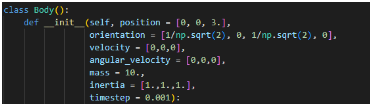
Vorteil - advantage.

Person A: Warum kaufst du immer bei diesem Laden ein?
Translation: Why do you always shop at this store?

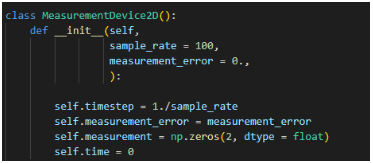
Person B: Weil es hier viele Vorteile gibt, wie z.B. kostenlose Lieferung und gute Qualität.
Translation: Because there are many advantages here, such as free delivery and good quality. 14:35

Or (**yet unsuccessfully :c**) train a neural network to analyze brain waves and create a *brain-computer interface* ([J. Vidal]) using commercial EEG headset like Emotiv

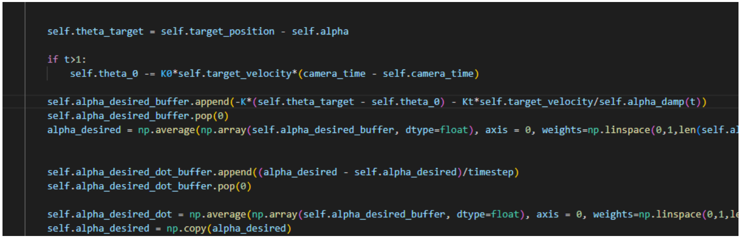




```
class Body():
    def __init__(self, position = [0, 0, 3.],
                 orientation = [1/np.sqrt(2), 0, 1/np.sqrt(2), 0],
                 velocity = [0,0,0],
                 angular_velocity = [0,0,0],
                 mass = 10.,
                 inertia = [1.,1.,1.],
                 timestep = 0.001):
```



```
class MeasurementDevice2D():
    def __init__(self,
                 sample_rate = 100,
                 measurement_error = 0.,
                 ):

        self.timestep = 1./sample_rate
        self.measurement_error = measurement_error
        self.measurement = np.zeros(2, dtype = float)
        self.time = 0
```



```
self.theta_target = self.target_position - self.alpha

if t>1:
    self.theta_0 -= K0*self.target_velocity*(camera_time - self.camera_time)

self.alpha_desired_buffer.append(-K*(self.theta_target - self.theta_0) - Kt*self.target_velocity/self.alpha_damp(t))
self.alpha_desired_buffer.pop(0)
alpha_desired = np.average(np.array(self.alpha_desired_buffer, dtype=float), axis = 0, weights=np.linspace(0,1,len(self.al


self.alpha_desired_dot_buffer.append((alpha_desired - self.alpha_desired)/timestep)
self.alpha_desired_dot_buffer.pop(0)

self.alpha_desired_dot = np.average(np.array(self.alpha_desired_buffer, dtype=float), axis = 0, weights=np.linspace(0,1,le
self.alpha_desired = np.copy(alpha_desired)
```

The end.