Example notebook for classification analysis

This notebook contains an example analysis of WatChMaL classification runs, including comparison to fiTQun reconstruction.

Plots are included of the training progression, ROC curves, and various plots of signal or background efficiency vs physical quantities.

Imports and setup

First change some display settings for the notebook

In [1]:
```
%load_ext autoreload
%autoreload 2
```

In [2]:
```
from IPython.display import display, HTML
display(HTML("<style>.container { width:100% !important; }</style>"))
display(HTML("<style>div.output_scroll { height: 44em; }</style>"))
```

External imports:

In [3]:
```
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
import scipy.optimize as opt
import sys
import h5py
import tabulate
```

In [4]:
```
# either add WatChMaL repository directory to PYTHONPATH environment vari
sys.path.append('/home/surajrai1900/WatChMaL')
sys.path.append('/home/surajrai1900/.local/bin')
```

Import WatChMaL analysis code

In [5]:
```
import analysis.utils.math as math
from analysis.utils.binning import get_binning, apply_binning
from analysis.utils.plotting import plot_legend
from analysis.classification import WatChMaLClassification, FiTQunClassif
from analysis.read import FiTQunOutput
```

```
/home/surajrai1900/WatChMaL/analysis/__init__.py:4: UserWarning: WARNIN
G: The git repository has uncommitted changes. Please commit changes bef
ore running WatChMaL code for proper version control
  print(f"Imported analysis code from WatChMaL repository with git versi
on: {get_git_version(os.path.dirname(__file__))}")
Imported analysis code from WatChMaL repository with git version: 0ed2fb
b-dirty
```

Preparing data

In [6]:
```
# list of particle names to use in filename
particle_names = ['gamma', 'e-', 'mu-', 'pi0'] # note that the order corr
```

Get true particle data from h5 file

In [7]:
```
# list of particle names to use in filename
particle_names = ['gamma', 'e-', 'mu-', 'pi0'] # note that the order corr
```

In [8]:
```
# get indices of test events
idxs_path = '/home/pdeperio/machine_learning/data/IWCD_mPMT_Short/index_l
test_idxs  =  np.load(idxs_path, allow_pickle=True)['test_idxs']
```

Open H5 data

In [9]:
```
# open h5 file and get data for test events
data_path = "/home/pdeperio/machine_learning/data/IWCD_mPMT_Short/IWCD_mP
h5_file = h5py.File(data_path, "r")
h5_angles     = np.array(h5_file['angles'])[test_idxs].squeeze()
h5_energies   = np.array(h5_file['energies'])[test_idxs].squeeze()
h5_positions  = np.array(h5_file['positions'])[test_idxs].squeeze()
h5_labels     = np.array(h5_file['labels'])[test_idxs].squeeze()
h5_root_files = np.array(h5_file['root_files'])[test_idxs].squeeze()
h5_event_ids  = np.array(h5_file['event_ids'])[test_idxs].squeeze()
h5_vetos      = np.array(h5_file['veto'])[test_idxs].squeeze()
events_hits_index = np.append(h5_file['event_hits_index'], h5_file['hit_p
h5_nhits      = (events_hits_index[test_idxs+1] - events_hits_index[test_
```

In [10]:
```
h5_directions = math.direction_from_angles(h5_angles)
```

Get fiTQun data from files

In [11]:
```
# list of particle names to use in filename
particle_names = ['gamma', 'e-', 'mu-', 'pi0'] # note that the order corr
```

fiTQun data for the IWCD events is stored in separate files for each particle type. Here we get the set of fiTQun files and open them using the FiTQunOutput class, which provides many useful attributes and methods for accessing fiTQun outputs:

In [12]:
```
help(FiTQunOutput)
```

```
Help on class FiTQunOutput in module analysis.read:

class FiTQunOutput(builtins.object)
 |  FiTQunOutput(file_path)
 |
 |  Class for reading in results of fiTQun reconstruction. Documentation
of the outputs provided is mostly taken
 |  directly from the fiTQun readme file. See github.com/fiTQun/fiTQun
(access to private repository required) for more
 |  details.
 |
 |  Time-window information
 |  ----------------------
 |  The following attributes are provided for the fiTQun outputs of the
hit time clustering algorithm:
 |
 |  ================================================================
==================================
 |  Attribute name        fiTQun output     Description
 |  ================================================================
==================================
 |  n_timewindows         fqntwnd           Number of time windows (good
clusters) in this event
 |  timewindow            fqtwnd            Cluster index of the time wi
ndow(corresponds to cluster_ncand)
 |  timewindow_cluster    fqtwnd_iclstr     Number of peaks(sub-events)
in the time window
 |  timewindow_time       fqtwnd_prftt0     Pre-fitter vertex time
 |  timewindow_position   fqtwnd_prftpos    Pre-fitter vertex position
 |  timewindow_npeaks     fqtwnd_npeak      Time window start/end time
 |  timewindow_peak_time  fqtwnd_peakt0     Time of each sub-event in th
e time window
 |  timewindow_peakiness  fqtwnd_peakiness  Vertex goodness parameter ev
aluated at the peak position
 |  ================================================================
==================================
 |
 |  Sub-event information
 |  --------------------
 |  The following attributes are provided for the fiTQun outputs of the
sub-events of the hit time clustering algorithm:
 |
 |  ================================================================
==============================================
 |  Attribute name               fiTQun output  Description
 |  ================================================================
==============================================
 |  n_subevents                  fqnse          Total number of subevent
s in the event
 |  subevent_timewindow          fqitwnd        Index of the time window
to which the subevent belongs
 |  subevent_peak                fqipeak        Peak index within the ti
me window
 |  subevent_nhits               fqnhitpmt      Number of hits in each s
ubevent
 |  subevent_total_charge        fqtotq         Total charge in each sub
event
 |  subevent_0ring_total_charge  fq0rtotmu      Total predicted charge f
or the 0-ring hypothesis in each subevent -
 |                                              these variables are the
```

```
result of evaluating the likelihood function
 |                                                  with a particle that is
below Cherenkov threshold.
 |  subevent_0ring_nll           fq0rnll        -log(L) value for the 0-
ring hypothesis in each subevent
 |  subevent_n50                 fqn50          n50 - In the TOF-correct
ed hit time distribution, number of hits within
 |                                                  the 50ns time window cen
tred at vertex time (1R electron fit vertex is
 |                                                  used)
 |  subevent_q50                 fqq50          q50 - Total charge of hi
ts included in n50 above
 |  ================================================================
================================================
 |
 |  1-Ring fits
 |  -----------
 |  These variables are the result of the 1-ring fits. The first index i
s the subevent (1-ring fits are run on all
 |  subevents). The second index is the particle-hypothesis index (same
as apfit):
 |  0 = GAMMA, 1 = ELECTRON, 2 = MUON, 3 = PION, 4 = KAON, 5 = PROTON,
6 = CONE GENERATOR
 |  Currently, only the electron, muon, and pion (the upstream pion segm
ent) hypotheses are implemented.
 |
 |  The following attributes are provided for the fiTQun outputs of elec
tron and muon sub-fits of the 1-ring fit results
 |  for the first sub-event:
 |
 |  ================================================================
================================================
 |  Attribute name         fiTQun output         Description
 |  ================================================================
================================================
 |  electron_flag          fq1rpcflg [][0][1]    Flag to indicate whethe
r fiTQun believes the electron is exiting the ID
 |                                                  (<0 if MINUIT did not c
onverge)
 |  electron_momentum      fq1rmom   [][0][1]    Fit electron momentum
 |  electron_position      fq1rpos   [][0][1][]  Fit electron vertex (0=
X, 1=Y, 2=Z)
 |  electron_direction     fq1rdir   [][0][1][]  Fit electron direction
(0=X, 1=Y, 2=Z)
 |  electron_time          fq1rt0    [][0][1]    Fit electron creation t
ime
 |  electron_total_charge  fq1rtotmu [][0][1]    Electron best-fit total
predicted charge
 |  electron_nll           fq1rnll   [][0][1]    Electron best-fit -lnL
 |  muon_flag              fq1rpcflg [][0][2]    Flag to indicate whethe
r fiTQun believes the muon is exiting the ID
 |                                                  (<0 if MINUIT did not c
onverge)
 |  muon_momentum          fq1rmom   [][0][2]    Fit muon momentum
 |  muon_position          fq1rpos   [][0][2][]  Fit muon vertex (0=X, 1
=Y, 2=Z)
 |  muon_direction         fq1rdir   [][0][2][]  Fit muon direction (0=
X, 1=Y, 2=Z)
 |  muon_time               fq1rt0   [][0][2]    Fit muon creation time
 |  muon_total_charge      1rtotmu   [][0][2]    Muon best-fit total pre
```

```
dicted charge
 |  muon_nll               fq1rnll   [][0][2]    Muon best-fit -lnL
 |  ================================================================================
================================================
 |
 |  Pi0 fits
 |  --------
 |  Pi0 fits are only run on the first sub-event. Index 0 gives the stan
dard, unconstrained pi0 fit. (Index 1 is not
 |  filled currently)
 |  The following attributes are provided for the fiTQun outputs of the
unconstrained-mass sub-fit of the pi0 fit for
 |  the first sub-event:
 |
 |  ================================================================================
========================================
 |  Attribute name               fiTQun output           Description
 |  ================================================================================
========================================
 |  pi0_flag                     fqpi0pcflg     [][0]    (PCflg for phot
on 1) + 2*(PCflg for photon 2)
 |  pi0_momentum                 fqpi0momtot    [][0]    Fit momentum of
the pi0
 |  pi0_position                 fqpi0pos       [][0][]  Fit vertex posi
tion
 |  pi0_direction                fqpi0dirtot    [][0][]  Fit direction o
f the pi0
 |  pi0_time                     fqpi0t0        [][0]    Fit pi0 creatio
n time
 |  pi0_total_charge             fqpi0totmu     [][0]    Best fit total
predicted charge
 |  pi0_nll                      fqpi0nll       [][0]    Best fit -log(L
ikelihood)
 |  pi0_mass                     fqpi0mass      [][0]    Fit pi0 mass (a
lways 134.9766 for constrained mass fit)
 |  pi0_gamma1_momentum          fqpi0mom1      [][0]    Fit momentum of
first photon
 |  pi0_gamma2_momentum          fqpi0mom2      [][0]    Fit momentum of
second photon
 |  pi0_gamma1_direction         fqpi0dir2      [][0][]  Fit direction o
f the first photon
 |  pi0_gamma2_direction         fqpi0dir2      [][0][]  Fit direction o
f the second photon
 |  pi0_gamma1_conversion_length fqpi0dconv2    [][0]    Fit conversion
length for the first photon
 |  pi0_gamma2_conversion_length fqpi0dconv2    [][0]    Fit conversion
length for the second photon
 |  pi0_gamma_opening_angle      fqpi0photangle [][0]    Fit opening ang
le between the photons
 |  ================================================================================
========================================
 |
 |  Multi-Ring fits
 |  ---------------
 |  These are the results of the Multi-Ring (MR) fits. The number of exe
cuted multi-ring fits depends on the event
 |  topology, and the first index specifies different fit results. (Inde
x 0 is the best-fit result.)
 |  Each fit result is assigned a unique fit ID which tells the type of
the fit(see fiTQun.cc for more details):
```

```
 |
 |  8-digit ID "N0...ZYX" :
 |      These are the raw MR fitter output, in which a ring is either an
electron or a pi+. The most significant digit
 |      "N" is the number of rings(1-6), and X, Y and Z are the particle
type(as in 1R fit, "1" for e, "3" for pi+) of
 |      the 1st, 2nd and 3rd ring respectively. Negative fit ID indicate
s that the ring which is newly added in the fit
 |      is determined as a fake ring by the fake ring reduction algorith
m.
 |
 |  9-digit ID "1N0...ZYX" :
 |      These are the results after the fake ring reduction is applied o
n the raw MR fit results above with ID
 |      "N0...ZYX". Rings are re-ordered according to their visible ener
gy, and one needs refer to the fqmrpid variable
 |      for the particle type of each ring, not the fit ID.
 |
 |  9-digit ID "2N0...ZYX" :
 |      These are the results after the fake ring merging and sequential
re-fitting are applied on the post-reduction
 |      result "1N0...ZYX". PID of a ring can change after the re-fit, a
nd muon hypothesis is also applied on the most
 |      energetic ring.
 |
 |  9-digit ID "3N0...ZYX" :
 |      These are the results after simultaneously fitting the longitudi
nal vertex position and the visible energy of
 |      all rings, on the post-refit result "2N0...ZYX".(Still experimen
tal)
 |
 |  9-digit ID "8NX000000" :
 |      When the best-fit hypothesis has more than one ring, the negativ
e log-likelihood values for each ring (N) and
 |      PID hypothesis (X) can be obtained using these results. For exam
ple, to compare the likelihood for the pion and
 |      electron hypotheses of the second ring, the IDs "813000000" and
"811000000" could be used.
 |
 |  The following attributes are provided for the fiTQun outputs of the
multi-ring fits:
 |
 |  ========================================================================
================================================
 |  Attribute name               fiTQun output  Description
 |  ========================================================================
================================================
 |  n_multiring_fits             fqnmrfit       Number of MR fit results
that are available
 |  multiring_fit_id             fqmrifit       Fit ID of each MR fit re
sult
 |  multiring_n_rings            fqmrnring      Number of rings for this
fit [1-6]
 |  multiring_flag               fqmrpcflg      <0 if MINUIT did not con
verge during the fit
 |  multiring_pid                fqmrpid        Particle type index for
each ring in the fit (Same convention as 1R fit)
 |  multiring_momentum           fqmrmom        Fit momentum of each rin
g
 |  multiring_position           fqmrpos        Fit vertex position of e
```

```
ach ring
 |  multiring_direction          fqmrdir        Fit direction of each ri
ng
 |  multiring_time               fqmrt0         Fit creation time of eac
h ring
 |  multiring_total_charge       fqmrtotmu      Best-fit total predicted
charge
 |  multiring_nll                fqmrnll        Best-fit -lnL
 |  multiring_conversion_length  fqmrdconv      Fit conversion length of
each ring(always "0" in default mode)
 |  multiring_energy_loss        fqmreloss      Energy lost in the upstr
eam track segment(for upstream tracks only)
 |  ================================================================
================================================
 |
 |  Multi-Segment Muon fits
 |  ----------------------
 |  These are the results of the Multi-Segment (M-S) muon fits. By defau
lt, the stand-alone M-S fit (first index="0") is
 |  applied on every event, and if the most energetic ring in the best-f
it MR fit is a muon, the muon track is re-fitted
 |  as a M-S track. (first index="1")
 |  The following attributes are provided for the fiTQun outputs of the
M-S fits:
 |
 |  ================================================================
================================================
 |  Attribute name             fiTQun output  Description
 |  ================================================================
================================================
 |  n_multisegment_fits        fqmsnfit       Number of Multi-Segment fi
t results that are available
 |  multisegment_flag          fqmspcflg      <0 if MINUIT did not conve
rge during the fit
 |  multisegment_n_segments    fqmsnseg       Number of track segments i
n the fit
 |  multisegment_pid           fqmspid        Particle type of the M-S t
rack (always "2")
 |  multisegment_fit_id        fqmsifit       Fit ID of the MR fit that
seeded this fit("1" for the stand-alone M-S fit)
 |  multisegment_ring          fqmsimer       Index of the ring to which
the M-S track corresponds in the seeding MR fit
 |  multisegment_momentum      fqmsmom        Fit initial momentum of ea
ch segment
 |  multisegment_position      fqmspos        Fit vertex position of eac
h segment
 |  multisegment_direction     fqmsdir        Fit direction of each segm
ent
 |  multisegment_time          fqmst0         Fit creation time of each
segment
 |  multisegment_total_charge  fqmstotmu      Best-fit total predicted c
harge
 |  multisegment_nll           fqmsnll        Best-fit -lnL
 |  multisegment_energy_loss   fqmseloss      Energy lost in each segmen
t
 |  ================================================================
================================================
 |
 |  Proton decay: p -> K+ nu; K+ -> mu+ nu; "prompt gamma method" fit
 |  ----------------------------------------------------------------
```

```
 |  These are the results of the PDK_MuGamma fit, dedicated to proton de
cay searches. Although there are two available
 |  fit results for each quantity, only the first is used (e.g. fqpmgmom
1[0])
 |  The following attributes are provided for the fiTQun outputs of the
PDK fit:
 |
 |  ========================================================================
======
 |  Attribute name       fiTQun output  Description
 |  ========================================================================
======
 |  pdk_flag             fqpmgpcflg     (PCflg for muon) + 2*(PCflg for
gamma)
 |  pdk_muon_momentum    fqpmgmom1      Best-fit muon momentum
 |  pdk_muon_position    fqpmgpos1      Best-fit muon position
 |  pdk_muon_direction   fqpmgdir1      Best-fit muon direction
 |  pdk_muon_time        fqpmgt01       Best-fit muon time
 |  pdk_gamma_momentum   fqpmgmom2      Best-fit gamma momentum
 |  pdk_gamma_position   fqpmgpos2      Best-fit gamma position
 |  pdk_gamma_direction  fqpmgdir2      Best-fit gamma direction
 |  pdk_gamma_time       fqpmgt02       Best-fit gamma time
 |  pdk_total_charge     fqpmgtotmu     Best-fit total predicted charge
 |  pdk_nll              fqpmgnll       Best-fit negative log-likelihood
 |  ========================================================================
======
 |
 |  Methods defined here:
 |
 |  __init__(self, file_path)
 |      Create an object holding results of a fiTQun reconstruction run,
given path to the output root file.
 |
 |      Parameters
 |      ----------
 |      file_path: str
 |          Path the fiTQun output root file
 |
 |  ------------------------------------------------------------------------
--
 |  Readonly properties defined here:
 |
 |  electron_direction
 |      Single electron-like ring fit direction (X, Y, Z)
 |
 |  electron_flag
 |      Flag to indicate whether fiTQun believes the electron is exiting
the ID(<0 if MINUIT did not converge)
 |
 |  electron_momentum
 |      Single electron-like ring fit momentum
 |
 |  electron_nll
 |      Single electron-like ring best-fit -lnL
 |
 |  electron_position
 |      Single electron-like ring fit vertex (X, Y, Z)
 |
 |  electron_time
 |      Single electron-like ring fit particle creation time
```

```
 |
 |  electron_total_charge
 |      Single electron-like ring best-fit total predicted charge
 |
 |  muon_direction
 |      Single muon-like ring fit direction (X, Y, Z)
 |
 |  muon_flag
 |      Flag to indicate whether fiTQun believes the muon is exiting the
ID(<0 if MINUIT did not converge)
 |
 |  muon_momentum
 |      Single muon-like ring fit momentum
 |
 |  muon_nll
 |      Single muon-like ring best-fit -lnL
 |
 |  muon_position
 |      Single muon-like ring fit vertex (X, Y, Z)
 |
 |  muon_time
 |      Single muon-like ring fit particle creation time
 |
 |  muon_total_charge
 |      Single muon-like ring best-fit total predicted charge
 |
 |  pi0_direction
 |      Fit direction of the pi0
 |
 |  pi0_flag
 |      (PCflg for photon 1) + 2*(PCflg for photon 2)
 |
 |  pi0_gamma1_conversion_length
 |      Fit conversion length for the first photon
 |
 |  pi0_gamma1_direction
 |      Fit direction of the first photon
 |
 |  pi0_gamma1_momentum
 |      Fit momentum of first photon
 |
 |  pi0_gamma2_conversion_length
 |      Fit conversion length for the second photon
 |
 |  pi0_gamma2_direction
 |      Fit direction of the second photon
 |
 |  pi0_gamma2_momentum
 |      Fit momentum of second photon
 |
 |  pi0_gamma_opening_angle
 |      Fit opening angle between the photons
 |
 |  pi0_mass
 |      Fit pi0 mass (always 134.9766 for constrained mass fit)
 |
 |  pi0_momentum
 |      Fit momentum of the pi0
 |
 |  pi0_nll
```

```
 |      pi0 best-fit -log(Likelihood)
 |
 |  pi0_position
 |      pi0 fit vertex position
 |
 |  pi0_time
 |      Fit pi0 creation time
 |
 |  pi0_total_charge
 |      pi0 best-fit total predicted charge
 |
 |  ----------------------------------------------------------------------
--
 |  Data descriptors defined here:
 |
 |  __dict__
 |      dictionary for instance variables (if defined)
 |
 |  __weakref__
 |      list of weak references to the object (if defined)
```

In [13]:
```
# get the data from the fitqun files
fq_path = "/home/pdeperio/machine_learning/data/IWCD_mPMT_Short/fiTQun/"
particle_order = [particle_names[i] for i in h5_labels[::1200000]] # Ther
fq_files = [fq_path + f"IWCD_mPMT_Short_{p}_E0to1000MeV_unif-pos-R400-y30
fq = FiTQunOutput(fq_files)
```

Find the fiTQun events corresponding to the test set

For the IWCD mPMT e/mu/gamma/pi0 data, there are events missing from the h5 file (used for training and testing) that are not missing from the fiTQun set, so we need to find how to match them up properly. We use the following to match them up:

- Each event in the h5 file has a `label` (particle type), a `root_file` and a `event_id`
- The events in the h5 file are ordered by `label` then `root_file` then `event_id`
- The fiTQun files have all events for one particle type, ordered by `root_file`, then by `event_id`
- Each `root_file` originally has exactly 3000 events in the fiTQun file
- The `event_id`s in for each root file in the fiTQun file go consecutively from 0 to 2999
- The h5 file has a subset of the events in the fiTQun files, and has at least one event from every `root_file`

So we could loop through the h5 file events, check the `label` to find which particle it is to choose the corresponding fitqun file, check the `root_file`, if it's the same as the previous then we compare to previous `event_id` and if there's a jump then we skip [new `event_id` minus old `event_id` +1] events in the fiTQun files. If it's a new `root_file` then we skip [new `event_id` plus 3000 minus old `event_id` +1].

Or, we do it more efficiently by adding up index offsets by particle type, root file and event_id, because we know:

- There are 400 `root_files` included in the test set for each particle
- There are 3000 `event_id`s running from 0 to 2999 for each `root_file`

So, if we index the `root_files` from 0 to 399 for particle 0, 400 to 799 for particle 1, and so on, then the offset is just `3000*[root_file_index] + [event_id]`

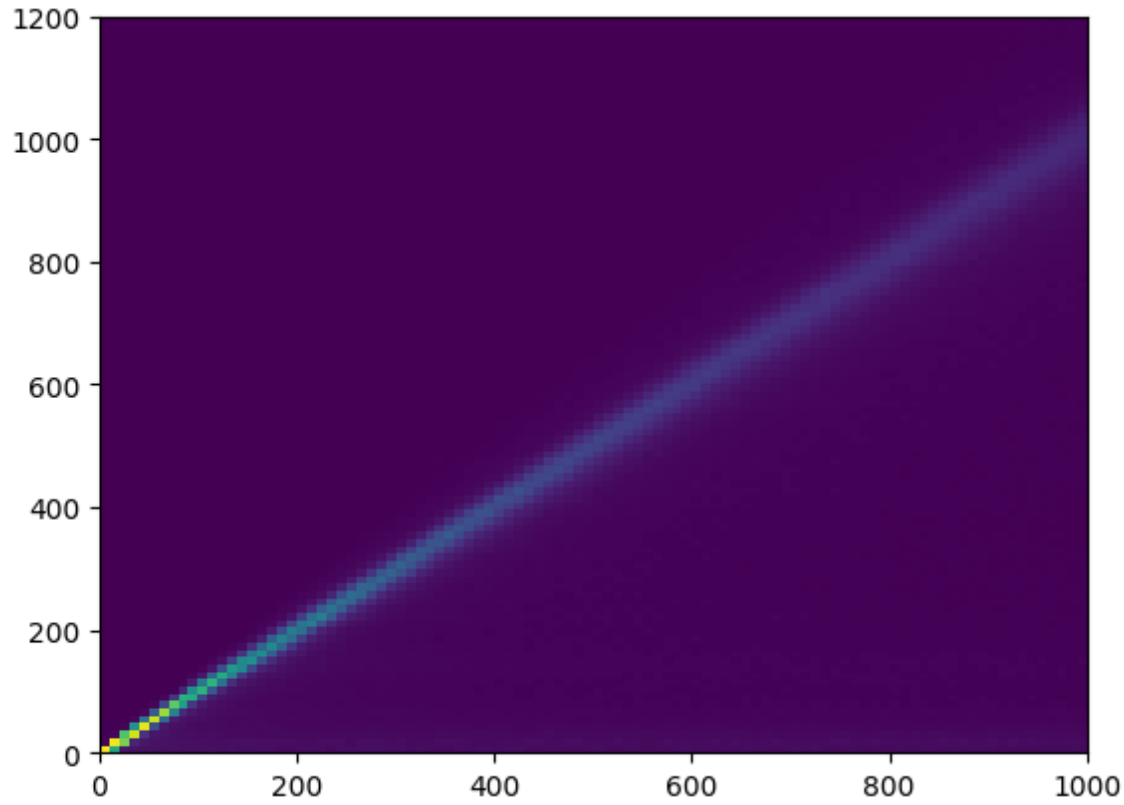
In [14]:
```
# create dictionary with keys of the unique root_files
root_file_index = dict.fromkeys(h5_root_files)
```

In [15]:
```
# update the values so each root file key has a value from 0 to the numbe
root_file_index.update((k, i) for i, k in enumerate(root_file_index))
```

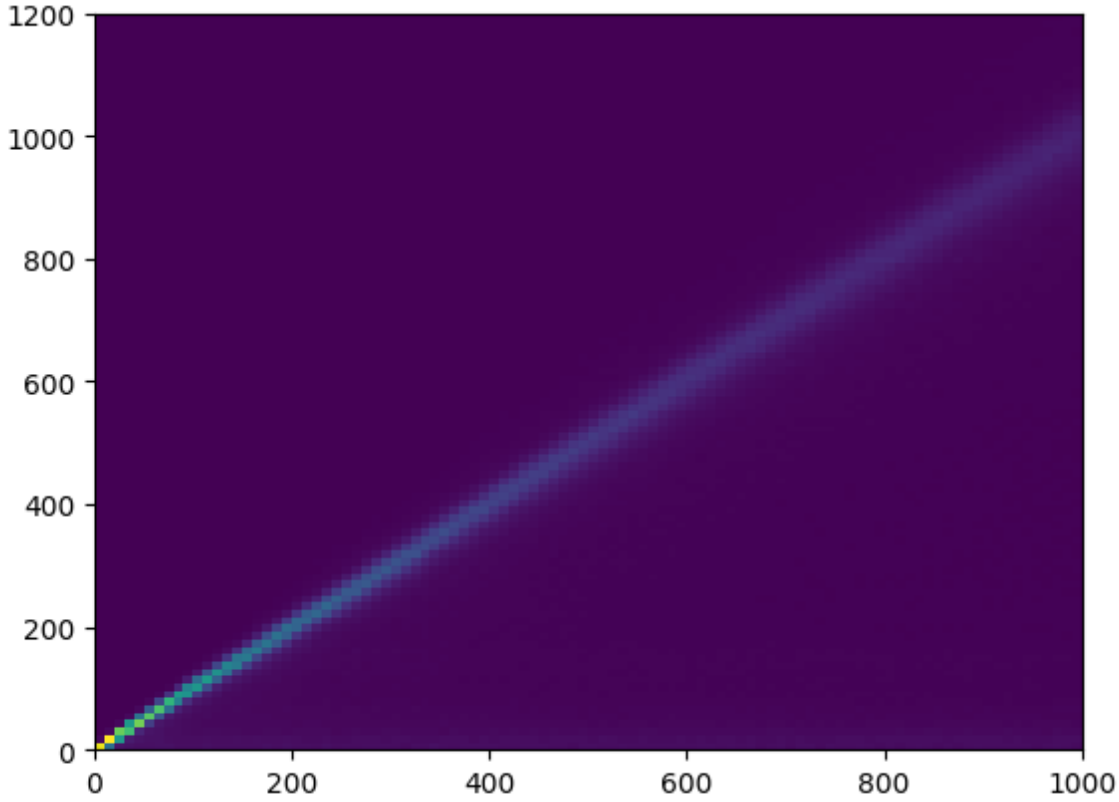
In [16]:
```
# create array of root_file_indices of each event in the h5_root_files ar
root_file_indices = np.vectorize(root_file_index.__getitem__)(h5_root_fil
# create the array of offsets from the root_file_indices and the h5_event
h5_fq_offsets = 3000*root_file_indices + h5_event_ids
```

In [17]:
```
# Get the fiTQun outputs corresponding to the h5 file's events
h5_flag_e = np.array(fq.electron_flag[h5_fq_offsets])
h5_flag_mu = np.array(fq.muon_flag[h5_fq_offsets])
h5_reco_mom_e = np.array(fq.electron_momentum[h5_fq_offsets])
h5_reco_mom_mu = np.array(fq.muon_momentum[h5_fq_offsets])
h5_reco_mom_pi0 = np.array(fq.pi0_momentum[h5_fq_offsets])
```

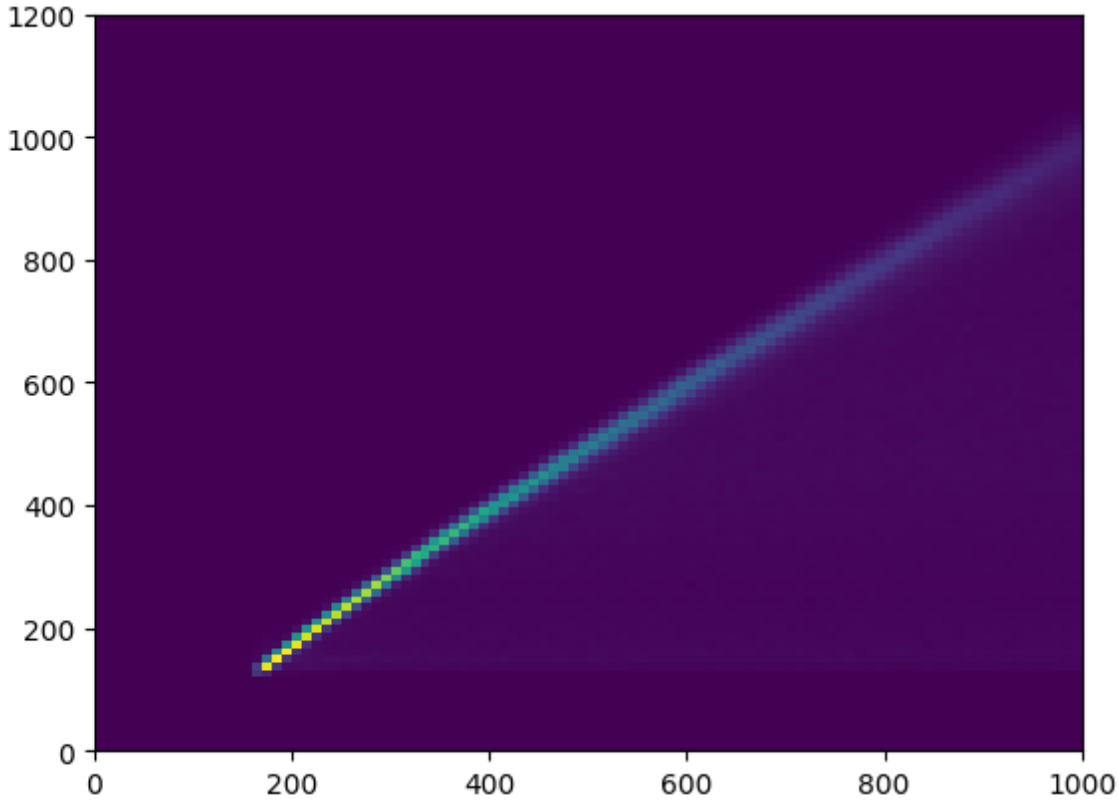
In [18]:
```
# check gammas seem to be matched right
p = plt.hist2d(h5_energies[h5_labels==0], h5_reco_mom_e[h5_labels==0], bi
```



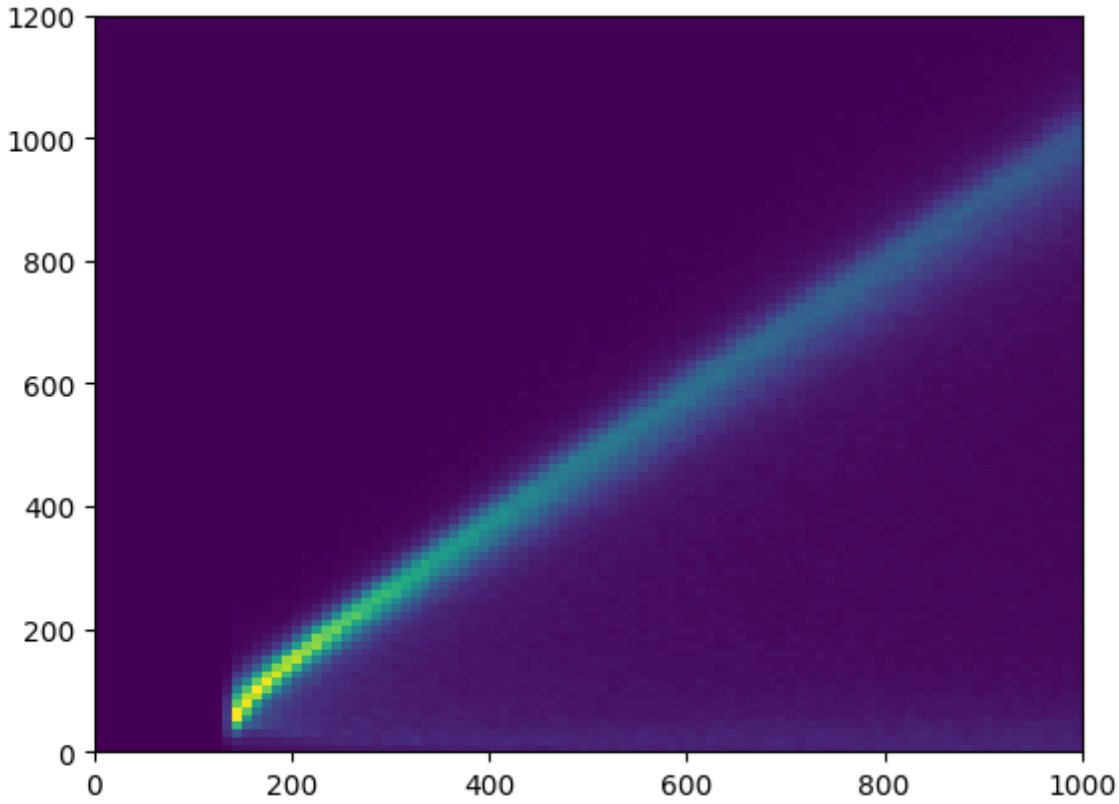
In [19]:
```
# check electrons seem to be matched right
p = plt.hist2d(h5_energies[h5_labels==1], h5_reco_mom_e[h5_labels==1], bi
```



In [20]:
```
# check muons seem to be matched right
p = plt.hist2d(h5_energies[h5_labels==2], h5_reco_mom_mu[h5_labels==2], b
```



In [21]:
```
# check pi0s seem to be matched right
p = plt.hist2d(h5_energies[h5_labels==3], h5_reco_mom_pi0[h5_labels==3],
```



Plotting fiTQun and ResNet performance

Set up plotting style

In [22]:
```
font = {'family' : 'DejaVu Sans',
        'weight' : 'normal',
        'size'   : 28}
matplotlib.rc('font', **font)
matplotlib.rcParams['figure.figsize'] = (12, 9)
matplotlib.rcParams["figure.autolayout"] = True
```

Make some cuts to select the events of interest

In [23]:
```
h5_towall = math.towall(h5_positions, h5_angles)
h5_dwall = math.dwall(h5_positions)
h5_momentum = math.momentum_from_energy(h5_energies, h5_labels)

reco_cut = (h5_flag_e == 0) & (h5_flag_mu == 0)
od_veto_cut = (h5_vetos == 0)
electron_containment_cut = ((h5_towall/100) > (0.63*np.log(h5_momentum) -
dwall_cut = h5_dwall > 50
towall_cut = h5_towall > 200
nhit_cut = h5_nhits > 25

h5_gammas = (h5_labels==0)
h5_electrons = (h5_labels==1)
h5_muons = (h5_labels==2)

# select the true electron and muon events that pass the cuts
```

```
basic_cuts = ((h5_electrons | h5_muons)
              & od_veto_cut
              & nhit_cut
             )
all_cuts = (basic_cuts
            & electron_containment_cut
            & reco_cut
            & dwall_cut
#             & towall_cut
            )

eg_basic_cuts = ((h5_electrons | h5_gammas)
              & od_veto_cut
              & nhit_cut
             )
eg_all_cuts = (eg_basic_cuts
            & electron_containment_cut
            & reco_cut
            & dwall_cut
#             & towall_cut
            )
```

Bin events in various quantities

Here we use the binning functions that allow us to bin events by various quantities:

In [24]:
```
help(get_binning)
```

```
Help on function get_binning in module analysis.utils.binning:

get_binning(x, bins=None, minimum=None, maximum=None, width=None)
    Finds the indices of the bins to which each value in input array bel
ongs, for a set of bins specified either as an
    array of bin edges, number of bins or bin width

    Parameters
    ----------
    x: array_like
        Input array to be binned.
    bins: array_like, optional
        If `bins` is an int, it defines the number of equal-width bins i
n the range (200, by default). If `bins` is an
        array, it is the array of bin edges and must be 1-dimensional an
d monotonic.
    minimum: int or real, optional
        Lowest bin lower edge (by default use minimum value in `x`). Not
used if `bins` is an ndarray of bin edges.
    maximum: int or real, optional
        Highest bin upper edge (by default use minimum value in `x`). No
t used if `bins` is an ndarray of bin edges.
    width: int or real, optional
        Width of bins to generate equal width bins if `bins` is None

    Returns
    -------
    bins: np.ndarray
        array of bin edges
    indices: np.ndarray
        output array of indices, of same shape as x
```

In [25]:
```
mom_binning = get_binning(h5_momentum, 20, 0, 1000)
mom_binning_mu = get_binning(h5_momentum, 17, 150, 1000)
reco_mom_e_binning = get_binning(h5_reco_mom_e, 20, 0, 1000)
cos_zenith_binning = get_binning(np.cos(h5_angles[:,0]), 20, -1, 1)
azimuth_binning = get_binning(h5_angles[:,1]*180/np.pi, 20, -180, 180)
dwall_binning = get_binning(h5_dwall, 22, 50, 300)
towall_binning = get_binning(h5_towall, 30, 50, 800)
```

Set up plotting style

In [26]:
```
font = {'family' : 'DejaVu Sans',
        'weight' : 'normal',
        'size'   : 28}
matplotlib.rc('font', **font)
matplotlib.rcParams['figure.figsize'] = (12, 9)
matplotlib.rcParams["figure.autolayout"] = True
```

Electron vs muon

Load results

Load ResNet and fiTQun results

Here we set up instances of the two classification classes, for WatChMaL and fiTQun outputs:

In [27]:
```
help(WatChMaLClassification)
help(FiTQunClassification)
```

```
Help on class WatChMaLClassification in module analysis.classification:

class WatChMaLClassification(ClassificationRun, analysis.read.WatChMaLOu
tput)
 |  WatChMaLClassification(directory, run_label, true_labels=None, indic
es=None, selection=None, **plot_args)
 |
 |  Class to hold results of a WatChMaL classification run
 |
 |  Method resolution order:
 |      WatChMaLClassification
 |      ClassificationRun
 |      analysis.read.WatChMaLOutput
 |      abc.ABC
 |      builtins.object
 |
 |  Methods defined here:
 |
 |  __init__(self, directory, run_label, true_labels=None, indices=None,
selection=None, **plot_args)
 |      Constructs the object holding the results of a WatChMaL classifi
cation run.
 |
 |      Parameters
 |      ----------
 |      directory: str
 |          Top-level output directory of a WatChMaL classification run.
 |      run_label: str
 |          Label to describe this set of results to use in plot legend
s, etc.
 |      true_labels: array_like of int, optional
 |          Array of true labels for the events in these classification
results
 |      indices: array_like of int, optional
 |          Array of indices of events to select out of the indices outp
ut by WatChMaL (by default use all events sorted
 |          by their indices).
 |      selection: index_expression, optional
 |          Selection to apply to the set of events to only use a subset
of all events when plotting results, etc.
 |          By default, use all results.
 |      plot_args: optional
 |          Additional arguments to pass to plotting functions, used to
set the style when plotting these results
 |          together with other runs' results.
 |
 |  discriminator(self, signal_labels, background_labels)
 |      Return a discriminator with appropriate scaling of softmax value
s from multi-class training, given the set of
 |      signal and background class labels. For each event, the discrimi
nator is the sum the signal softmax values
 |      normalised by the sum of signal and background softmax values.
 |
 |      Parameters
 |      ----------
 |      signal_labels: int or sequence of ints
 |          Set of labels corresponding to signal classes. Can be either
a single label or a sequence of labels.
 |      background_labels: int or sequence of ints
 |          Set of labels corresponding to background classes. Can be ei
```

```
ther a single label or a sequence of labels.
 |
 |      Returns
 |      -------
 |      np.ndarray
 |          One-dimensional array of discriminator values, with length e
qual to the number of events in this run.
 |
 |  plot_training_progression(self, plot_best=True, y_loss_lim=None, fig
_size=None, title=None, legend='center right')
 |      Plot the progression of training and validation loss and accurac
y from the run's logs
 |
 |      Parameters
 |      ----------
 |      plot_best: bool, optional
 |          If true (default), plot points indicating the best validatio
n loss and accuracy
 |      y_loss_lim: (int, int), optional
 |          Range for the loss y-axis. By default, the range will expand
to show all loss values in the logs.
 |      fig_size: (float, float), optional
 |          Size of the figure
 |      title: str, optional
 |          Title of the figure. By default, do not plot a title.
 |      legend: str, optional
 |          Position to plot the legend. By default, the legend is place
d in the center right. For no legend use `None`.
 |
 |      Returns
 |      -------
 |      matplotlib.figure.Figure
 |      matplotlib.axes.Axes
 |
 |  read_training_log_from_csv(self, directory)
 |      Read the training progression logs from the given directory.
 |
 |      Parameters
 |      ----------
 |      directory: str
 |          Path to the directory of the training run.
 |
 |      Returns
 |      -------
 |      np.ndarray
 |          Array of train epoch values for each entry in the training p
rogression log.
 |      np.ndarray
 |          Array of train loss values for each entry in the training pr
ogression log.
 |      np.ndarray
 |          Array of train accuracy values for each entry in the trainin
g progression log.
 |      np.ndarray
 |          Array of validation epoch values for each entry in the train
ing progression log
 |      np.ndarray
 |          Array of validation loss values for each entry in the traini
ng progression log
 |      np.ndarray
```

```
 |          Array of validation accuracy values for each entry in the tr
aining progression log
 |      np.ndarray
 |          Array of boolean values indicating whether each entry had th
e best validation loss so far in the training
 |          progression log
 |
 |  ----------------------------------------------------------------------
--
 |  Readonly properties defined here:
 |
 |  softmaxes
 |      Array of softmax outputs
 |
 |  train_log_accuracy
 |      Array of train accuracy values for each entry in the training pr
ogression log.
 |
 |  val_log_accuracy
 |      Array of validation accuracy values for each entry in the traini
ng progression log.
 |
 |  ----------------------------------------------------------------------
--
 |  Data and other attributes defined here:
 |
 |  __abstractmethods__ = frozenset()
 |
 |  ----------------------------------------------------------------------
--
 |  Methods inherited from ClassificationRun:
 |
 |  cut_with_constant_binned_efficiency(self, signal_labels, background_
labels, efficiency, binning, selection=None, select_labels=None, return_
thresholds=False)
 |      Generate array of boolean values indicating whether each event p
asses a cut defined such that, in each bin of
 |      some binning of the events, a constant proportion of the selecte
d events pass the cut.
 |      After taking the subset of `discriminator_values` defined by `se
lection`, in each bin of `binning` the threshold
 |      discriminator value is found such that the proportion that are a
bove the threshold is equal to `efficiency`.
 |      These cut thresholds are then used to apply the cut to all event
s (not just those selected by `selection`) and
 |      an array of booleans is returned for whether each discriminator
value is above the threshold of its
 |      corresponding bin. The cut result is also stored for use in plot
ting efficiency profiles.
 |
 |      Parameters
 |      ----------
 |      signal_labels: int or sequence of ints
 |          Set of labels corresponding to signal classes. Can be either
a single label or a sequence of labels.
 |      background_labels: int or sequence of ints
 |          Set of labels corresponding to background classes. Can be ei
ther a single label or a sequence of labels.
 |      efficiency: float
 |          The fixed efficiency to ensure in each bin.
```

```
 |      binning: (np.ndarray, np.ndarray)
 |          Array of bin edges and array of bin indices, returned from `
analysis.utils.binning.get_binning`.
 |      selection: indexing expression, optional
 |          Selection of the discriminator values to use in calculating
the thresholds applied by the cut in each bin
 |          (by default use the run's predefined selection, or all event
s if none is defined).
 |      select_labels: set of int, optional
 |          Set of true labels to select events to use in calculating th
e thresholds.
 |      return_thresholds: bool, optional
 |          If True, return also the array of cut thresholds calculated
for each bin.
 |
 |      Returns
 |      -------
 |      cut: np.ndarray of bool
 |          One-dimensional array, length of the total number of events,
indicating whether each event passes the cut.
 |      thresholds: np.ndarray of float, optional
 |          One-dimensional array giving the threshold applied by the cu
t to events in each bin.
 |
 |  cut_with_fixed_efficiency(self, signal_labels, background_labels, ef
ficiency, selection=None, select_labels=None, return_threshold=False)
 |      Generate array of boolean values indicating whether each event p
asses a cut defined such that a fixed proportion
 |      of the selected events pass the cut.
 |      After taking the subset of `discriminator_values` defined by `se
lection`, the threshold discriminator value is
 |      found such that the proportion that are above the threshold is e
qual to `efficiency`. This cut threshold is then
 |      used to apply the cut to all events (not just those selected by
`selection`) and an array of booleans is
 |      returned for whether each discriminator value is above the thres
hold of its corresponding bin. The cut result is
 |      also stored for use in plotting efficiency profiles.
 |
 |      Parameters
 |      ----------
 |      signal_labels: int or sequence of ints
 |          Set of labels corresponding to signal classes. Can be either
a single label or a sequence of labels.
 |      background_labels: int or sequence of ints
 |          Set of labels corresponding to background classes. Can be ei
ther a single label or a sequence of labels.
 |      efficiency: float
 |          The fixed efficiency.
 |      selection: indexing expression, optional
 |          Selection of the discriminator values to use in calculating
the threshold applied by the cut (by default use
 |          all values).
 |      select_labels: set of int
 |          Set of true labels to select events to use.
 |      return_threshold: bool, optional
 |          If True, return also the cut threshold.
 |
 |      Returns
 |      -------
```

```
 |      cut: np.ndarray of bool
 |          One-dimensional array the same length as `discriminator_valu
es` indicating whether each event passes the cut.
 |      threshold: float, optional
 |          The threshold applied by the cut.
 |
 |  plot_binned_efficiency(self, ax, binning, selection=None, select_lab
els=None, reverse=False, errors=False, x_errors=True, **plot_args)
 |      Plot binned efficiencies of the cut applied to the classificatio
n run on an existing set of axes.
 |      The cut values corresponding to booleans indicating whether each
event passes the cut are divided up into bins
 |      of some quantity according to `binning`, before calculating the
efficiency (proportion of events passing the
 |      cut) in each bin. A selection can be provided to use only a subs
et of all the values.
 |
 |      Parameters
 |      ----------
 |      ax: matplotlib.axes.Axes
 |          Axes to draw the plot.
 |      binning: (np.ndarray, np.ndarray)
 |          Array of bin edges and array of bin indices, returned from `
analysis.utils.binning.get_binning`.
 |      selection: indexing expression, optional
 |          Selection of the values to use in calculating the resolution
s (by default use all values).
 |      select_labels: set of int
 |          Set of true labels to select events to use.
 |      reverse: bool
 |          If True, reverse the cut to plot percentage of events failin
g the cut. By default, the percentage of events
 |          passing the cut is plotted
 |      errors: bool, optional
 |          If True, plot error bars calculated as the standard deviatio
n divided by sqrt(N) of the N values in the bin.
 |      x_errors: bool, optional
 |          If True, plot horizontal error bars corresponding to the wid
th of the bin, only if `errors` is also True.
 |      plot_args: optional
 |          Additional arguments to pass to the plotting function. Note
that these may be overridden by arguments
 |          provided in `runs`.
 |
 |  select_labels(self, select_labels, selection=None)
 |      Combine a selection of events with the additional requirement of
having chosen true labels.
 |
 |      Parameters
 |      ----------
 |      select_labels: set of int
 |          Set of true labels to select
 |      selection: index_expression, optional
 |          Selection over all events (by default use the run's predefin
ed selection)
 |
 |      Returns
 |      -------
 |      np.ndarray
 |          Array of indices that are both selected by `selection` and h
```

```
ave true label in `select_labels`
 |
 |  ----------------------------------------------------------------------
--
 |  Data descriptors inherited from ClassificationRun:
 |
 |  __dict__
 |      dictionary for instance variables (if defined)
 |
 |  __weakref__
 |      list of weak references to the object (if defined)
 |
 |  ----------------------------------------------------------------------
--
 |  Methods inherited from analysis.read.WatChMaLOutput:
 |
 |  get_outputs(self, name)
 |      Read the outputs resulting from the evaluation run of a WatChMaL
model.
 |
 |      Parameters
 |      ----------
 |      name: str
 |          name of the output to load
 |
 |      Returns
 |      -------
 |      np.ndarray
 |          Two dimensional array of predicted softmax values, where eac
h row corresponds to an event and each column
 |          contains the softmax values of a class.
 |
 |  read_training_log(self)
 |      Read the training progression logs for the run. If the run does
not have a training progression log, then logs
 |      are loaded from a run directory corresponding to a loaded pre-tr
ained state.
 |
 |      Returns
 |      -------
 |      tuple
 |          Tuple of arrays of training progression log values, see `rea
d_training_log_from_csv` for details.
 |
 |  ----------------------------------------------------------------------
--
 |  Readonly properties inherited from analysis.read.WatChMaLOutput:
 |
 |  train_log_epoch
 |      Array of train epoch values for each entry in the training progr
ession log.
 |
 |  train_log_loss
 |      Array of train loss values for each entry in the training progre
ssion log.
 |
 |  training_log
 |      Tuple of arrays of training progression log values, see `read_tr
aining_log_from_csv` for details.
 |
```

```
 |  val_log_best
 |      Array of boolean values indicating whether each entry had the be
st validation loss so far in the training
 |      progression log
 |
 |  val_log_epoch
 |      Array of validation epoch values for each entry in the training
progression log.
 |
 |  val_log_loss
 |      Array of validation loss values for each entry in the training p
rogression log.

Help on class FiTQunClassification in module analysis.classification:

class FiTQunClassification(ClassificationRun)
 |  FiTQunClassification(fitqun_output, run_label, true_labels=None, ind
ices=None, selection=None, particle_label_map=None, **plot_args)
 |
 |  Class to hold classification results of a fiTQun reconstruction run
 |
 |  Method resolution order:
 |      FiTQunClassification
 |      ClassificationRun
 |      abc.ABC
 |      builtins.object
 |
 |  Methods defined here:
 |
 |  __init__(self, fitqun_output, run_label, true_labels=None, indices=N
one, selection=None, particle_label_map=None, **plot_args)
 |      Create object containing classification results from a fiTQun re
construction run
 |
 |      Parameters
 |      ----------
 |      fitqun_output: analysis.read.FiTQunOutput
 |          Output from a fiTQun reconstruction run
 |      run_label: str
 |          Label to describe this set of results to use in plot legend
s, etc.
 |      true_labels: array_like of int, optional
 |          Array of true labels for the events in these classification
results
 |      indices: array_like of int, optional
 |          Array of indices of events to select out of the events in th
e fiTQun output (by default use all events).
 |      selection: index_expression, optional
 |          Selection to apply to the set of events to only use a subset
of all events when plotting results, etc.
 |          (by default use all events).
 |      particle_label_map: dict
 |          Dictionary mapping particle type names to label integers. By
default, use gamma:0, electron:1, muon:2, pi0:3
 |      plot_args: optional
 |          Additional arguments to pass to plotting functions, used to
set the style when plotting these results
 |          together with other runs' results.
 |
 |  discriminator(self, signal_labels, background_labels)
```

```
 |      Returns discriminator values given sets of labels representing t
he signal and background.
 |      For electron and/or gamma vs muon, use `electron_muon_discrimina
tor`.
 |      For electron vs pi0, use `electron_pi0_discriminator`.
 |      For electron vs gamma, use `electron_gamma_discriminator`.
 |      No other combination of signal and background has currently been
implemented for fiTQun results (other than
 |      swapping signal and background in any of the above cases, which
returns the discriminator multiplied by -1).
 |
 |      Parameters
 |      ----------
 |      signal_labels: int or sequence of ints
 |          Set of labels corresponding to signal classes. Can be either
a single label or a sequence of labels.
 |      background_labels: int or sequence of ints
 |          Set of labels corresponding to background classes. Can be ei
ther a single label or a sequence of labels.
 |
 |      Returns
 |      -------
 |      np.ndarray
 |          One-dimensional array of discriminator values, with length e
qual to the number of events in this run.
 |
 |  get_discriminator(self, discriminator)
 |      Helper function for defining a particular discriminator. If `dis
criminator` is a function, it should take the
 |      fiTQun output as its only argument and return the discriminator,
in which case the function called on this run's
 |      output is returned by this function. If `discriminator` is a str
ing, it should name an attribute of this class
 |      to use as the discriminator, in which case that attribute is ret
urned. In any other case the input is returned
 |      unchanged, for example if `discriminator` is already an array of
discriminator values.
 |
 |      Parameters
 |      ----------
 |      discriminator: callable or str or array_like of float
 |
 |      Returns
 |      -------
 |      ndarray of float
 |          Array of discriminator values
 |
 |  tune_nll_pi0mass_discriminator(self, pi0_efficiency=None, electron_e
fficiency=None, selection=None, binning=None, **opt_args)
 |      Tune the gradient of the cut line for a linear 2D cut in n(L_e)
- ln(L_pi0) and reconstructed pi0 mass.
 |      By default, optimize the gradient of the cut such that the Mann–
Whitney U test is minimised. This minimises the
 |      sum of the ranks of the pi0 discriminator values when ranked tog
ether with the electron discriminator values.
 |      If `pi0_efficiency` is given, then the gradient is optimized to
minimise the electron mis-PID when fixing a cut
 |      threshold that gives the desired pi0 efficiency.
 |      If `electron_efficiency` is given, then the gradient is optimize
d to minimise the pi0 mis-PID when fixing a cut
```

```
 |      threshold that gives the desired electron efficiency.
 |      If `binning` is provided, then the cut line gradient is tuned se
parately in each bin.
 |
 |      Parameters
 |      ----------
 |      pi0_efficiency: float, optional
 |          Fixed pi0 efficiency for which to minimise electron mis-PID
 |      electron_efficiency: float, optional
 |          Fixed electron efficiency for which to minimise pi0 mis-PID
 |      selection: index_expression, optional
 |          If provided, only consider selected events when optimising t
he cut. By default, use the run's pre-defined
 |          selection, if any.
 |      binning: (np.ndarray, np.ndarray), optional
 |          Result of `analysis.utils.binning.get_binning` to use to tun
e the cut separately in each bin. By default,
 |          the cut is tuned once for all events without binning.
 |      opt_args: optional
 |          Additional arguments to pass to `scipy.optimize.minimize_sca
lar`
 |
 |      Returns
 |      -------
 |      float or ndarray of floats
 |          The value of the optimal cut line gradient, or array of opti
mal cut line gradients in each bin if `binning`
 |          is provided.
 |
 |  ----------------------------------------------------------------------
--
 |  Readonly properties defined here:
 |
 |  electron_muon_discriminator
 |      Negative log-likelihood difference for electrons and muons: ln(L
_e) - ln(L_mu)
 |
 |  electron_pi0_nll_discriminator
 |      Electron vs pi0 log-likelihood difference: ln(L_e) - ln(L_pi0)
 |
 |  electron_pi0_nll_pi0mass_discriminator
 |      Linear 2D cut for electron vs pi0, in ln(L_e) - ln(L_pi0) and re
constructed pi0 mass
 |
 |  gamma_electron_discriminator
 |      Discriminator for gamma vs electron. The fiTQun gamma hypothesis
doesn't work well, so by default the electron
 |      vs gamma log-likelihood difference: ln(L_gamma) - ln(L_mu)
 |
 |  muon_electron_discriminator
 |      Negative log-likelihood difference for electrons and muons: ln(L
_mu) - ln(L_e)
 |
 |  pi0_electron_discriminator
 |      Discriminator for pi0 vs electron, by default the log-likelihood
difference: ln(L_pi0) - ln(L_e)
 |
 |  ----------------------------------------------------------------------
--
 |  Data descriptors defined here:
```

```
 |
 |  electron_gamma_discriminator
 |      Discriminator for electron vs gamma. The fiTQun gamma hypothesis
doesn't work well, so by default the muon vs
 |      electron log-likelihood difference: ln(L_mu) - ln(L_gamma)
 |
 |  electron_pi0_discriminator
 |      Discriminator for electron vs pi0, by default the log-likelihood
difference: ln(L_e) - ln(L_pi0)
 |
 |  nll_pi0mass_factor
 |      Gradient of the linear 2D cut in n(L_e) - ln(L_pi0) and reconstr
ucted pi0 mass
 |
 |  ----------------------------------------------------------------------
--
 |  Data and other attributes defined here:
 |
 |  __abstractmethods__ = frozenset()
 |
 |  ----------------------------------------------------------------------
--
 |  Methods inherited from ClassificationRun:
 |
 |  cut_with_constant_binned_efficiency(self, signal_labels, background_
labels, efficiency, binning, selection=None, select_labels=None, return_
thresholds=False)
 |      Generate array of boolean values indicating whether each event p
asses a cut defined such that, in each bin of
 |      some binning of the events, a constant proportion of the selecte
d events pass the cut.
 |      After taking the subset of `discriminator_values` defined by `se
lection`, in each bin of `binning` the threshold
 |      discriminator value is found such that the proportion that are a
bove the threshold is equal to `efficiency`.
 |      These cut thresholds are then used to apply the cut to all event
s (not just those selected by `selection`) and
 |      an array of booleans is returned for whether each discriminator
value is above the threshold of its
 |      corresponding bin. The cut result is also stored for use in plot
ting efficiency profiles.
 |
 |      Parameters
 |      ----------
 |      signal_labels: int or sequence of ints
 |          Set of labels corresponding to signal classes. Can be either
a single label or a sequence of labels.
 |      background_labels: int or sequence of ints
 |          Set of labels corresponding to background classes. Can be ei
ther a single label or a sequence of labels.
 |      efficiency: float
 |          The fixed efficiency to ensure in each bin.
 |      binning: (np.ndarray, np.ndarray)
 |          Array of bin edges and array of bin indices, returned from `
analysis.utils.binning.get_binning`.
 |      selection: indexing expression, optional
 |          Selection of the discriminator values to use in calculating
the thresholds applied by the cut in each bin
 |          (by default use the run's predefined selection, or all event
s if none is defined).
```

```
 |      select_labels: set of int, optional
 |          Set of true labels to select events to use in calculating th
e thresholds.
 |      return_thresholds: bool, optional
 |          If True, return also the array of cut thresholds calculated
for each bin.
 |
 |      Returns
 |      -------
 |      cut: np.ndarray of bool
 |          One-dimensional array, length of the total number of events,
indicating whether each event passes the cut.
 |      thresholds: np.ndarray of float, optional
 |          One-dimensional array giving the threshold applied by the cu
t to events in each bin.
 |
 |  cut_with_fixed_efficiency(self, signal_labels, background_labels, ef
ficiency, selection=None, select_labels=None, return_threshold=False)
 |      Generate array of boolean values indicating whether each event p
asses a cut defined such that a fixed proportion
 |      of the selected events pass the cut.
 |      After taking the subset of `discriminator_values` defined by `se
lection`, the threshold discriminator value is
 |      found such that the proportion that are above the threshold is e
qual to `efficiency`. This cut threshold is then
 |      used to apply the cut to all events (not just those selected by
`selection`) and an array of booleans is
 |      returned for whether each discriminator value is above the thres
hold of its corresponding bin. The cut result is
 |      also stored for use in plotting efficiency profiles.
 |
 |      Parameters
 |      ----------
 |      signal_labels: int or sequence of ints
 |          Set of labels corresponding to signal classes. Can be either
a single label or a sequence of labels.
 |      background_labels: int or sequence of ints
 |          Set of labels corresponding to background classes. Can be ei
ther a single label or a sequence of labels.
 |      efficiency: float
 |          The fixed efficiency.
 |      selection: indexing expression, optional
 |          Selection of the discriminator values to use in calculating
the threshold applied by the cut (by default use
 |          all values).
 |      select_labels: set of int
 |          Set of true labels to select events to use.
 |      return_threshold: bool, optional
 |          If True, return also the cut threshold.
 |
 |      Returns
 |      -------
 |      cut: np.ndarray of bool
 |          One-dimensional array the same length as `discriminator_valu
es` indicating whether each event passes the cut.
 |      threshold: float, optional
 |          The threshold applied by the cut.
 |
 |  plot_binned_efficiency(self, ax, binning, selection=None, select_lab
els=None, reverse=False, errors=False, x_errors=True, **plot_args)
```

```
 |      Plot binned efficiencies of the cut applied to the classificatio
n run on an existing set of axes.
 |      The cut values corresponding to booleans indicating whether each
event passes the cut are divided up into bins
 |      of some quantity according to `binning`, before calculating the
efficiency (proportion of events passing the
 |      cut) in each bin. A selection can be provided to use only a subs
et of all the values.
 |
 |      Parameters
 |      ----------
 |      ax: matplotlib.axes.Axes
 |          Axes to draw the plot.
 |      binning: (np.ndarray, np.ndarray)
 |          Array of bin edges and array of bin indices, returned from `
analysis.utils.binning.get_binning`.
 |      selection: indexing expression, optional
 |          Selection of the values to use in calculating the resolution
s (by default use all values).
 |      select_labels: set of int
 |          Set of true labels to select events to use.
 |      reverse: bool
 |          If True, reverse the cut to plot percentage of events failin
g the cut. By default, the percentage of events
 |          passing the cut is plotted
 |      errors: bool, optional
 |          If True, plot error bars calculated as the standard deviatio
n divided by sqrt(N) of the N values in the bin.
 |      x_errors: bool, optional
 |          If True, plot horizontal error bars corresponding to the wid
th of the bin, only if `errors` is also True.
 |      plot_args: optional
 |          Additional arguments to pass to the plotting function. Note
that these may be overridden by arguments
 |          provided in `runs`.
 |
 |  select_labels(self, select_labels, selection=None)
 |      Combine a selection of events with the additional requirement of
having chosen true labels.
 |
 |      Parameters
 |      ----------
 |      select_labels: set of int
 |          Set of true labels to select
 |      selection: index_expression, optional
 |          Selection over all events (by default use the run's predefin
ed selection)
 |
 |      Returns
 |      -------
 |      np.ndarray
 |          Array of indices that are both selected by `selection` and h
ave true label in `select_labels`
 |
 |  ----------------------------------------------------------------------
--
 |  Data descriptors inherited from ClassificationRun:
 |
 |  __dict__
 |      dictionary for instance variables (if defined)
```

```
 |
 |  __weakref__
 |      list of weak references to the object (if defined)
```

In [28]:
```
pid_runs = [
    (WatChMaLClassification("/home/surajrai1900/WatChMaL/outputs/2023-09-
     WatChMaLClassification("/home/pdeperio/machine_learning/data/IWCD_mP
    (WatChMaLClassification("/home/surajrai1900/WatChMaL/outputs/2023-09-
     WatChMaLClassification("/home/pdeperio/machine_learning/data/IWCD_mP
    FiTQunClassification(fq, "fiTQun (basic cuts)", h5_labels, h5_fq_offs
    FiTQunClassification(fq, "fiTQun (all cuts)", h5_labels, h5_fq_offset
]

directory = ["/home/surajrai1900/WatChMaL/outputs/2023-09-19/03-59-17/",
             "/home/pdeperio/machine_learning/data/IWCD_mPMT_Short/WatChMa
             "/home/surajrai1900/WatChMaL/outputs/2023-09-19/03-59-17/",
             "/home/pdeperio/machine_learning/data/IWCD_mPMT_Short/WatChMa
```

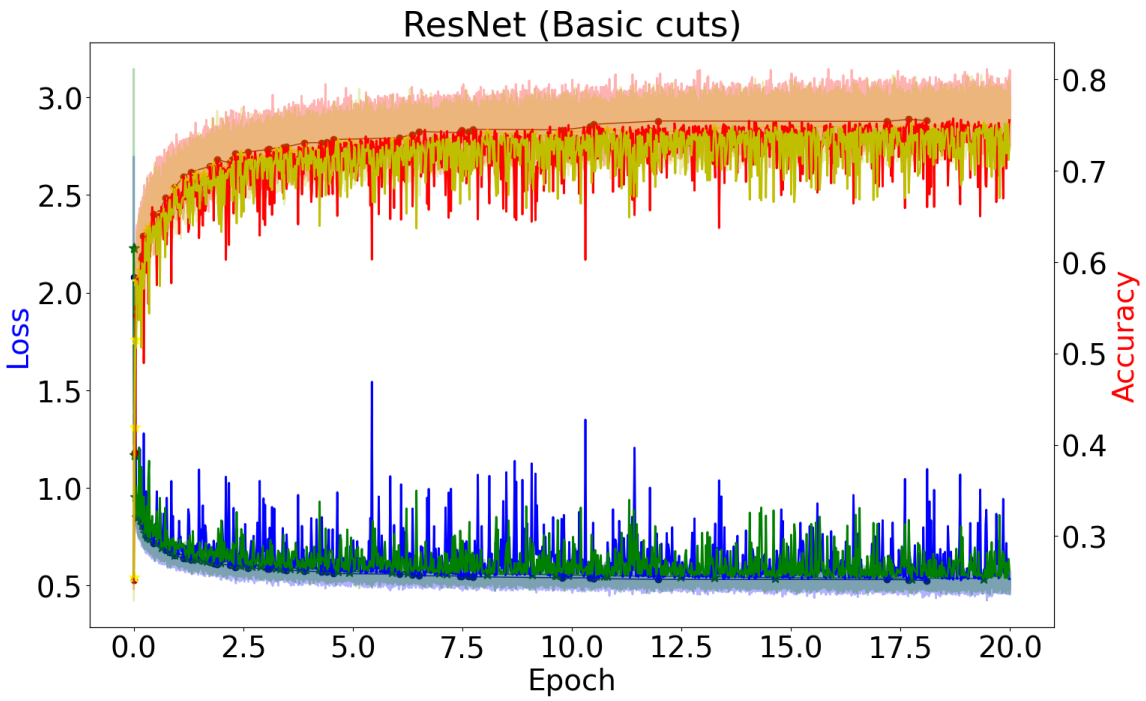
Plot training progression

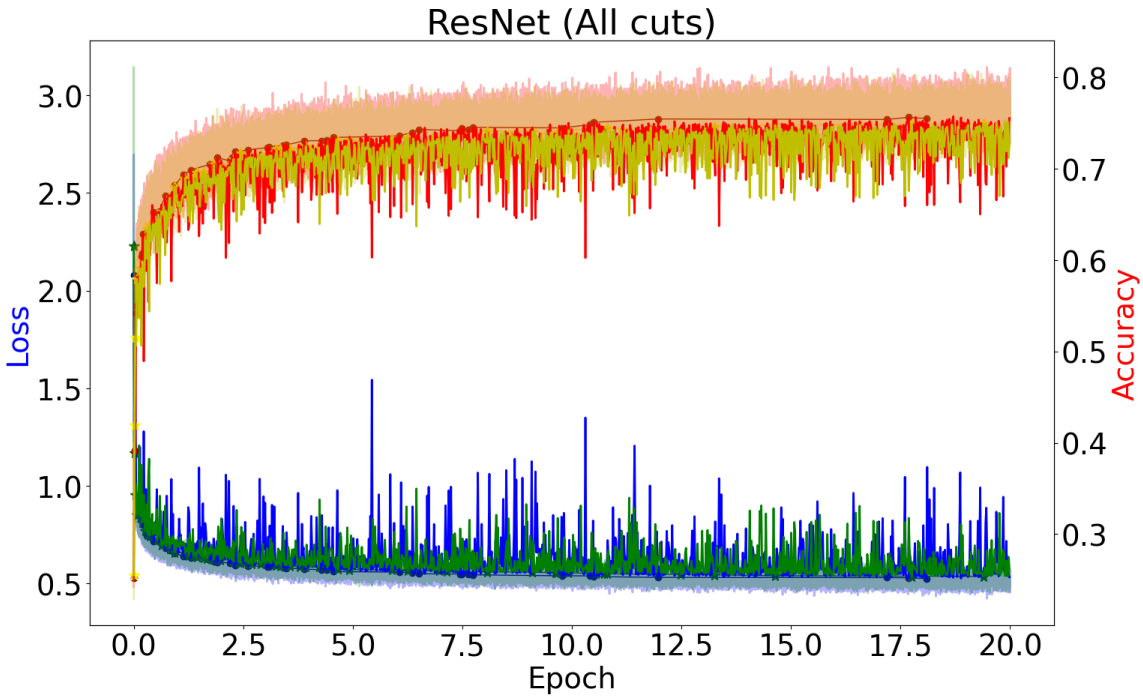
In [29]:
```
help(WatChMaLClassification.plot_training_progression)
```

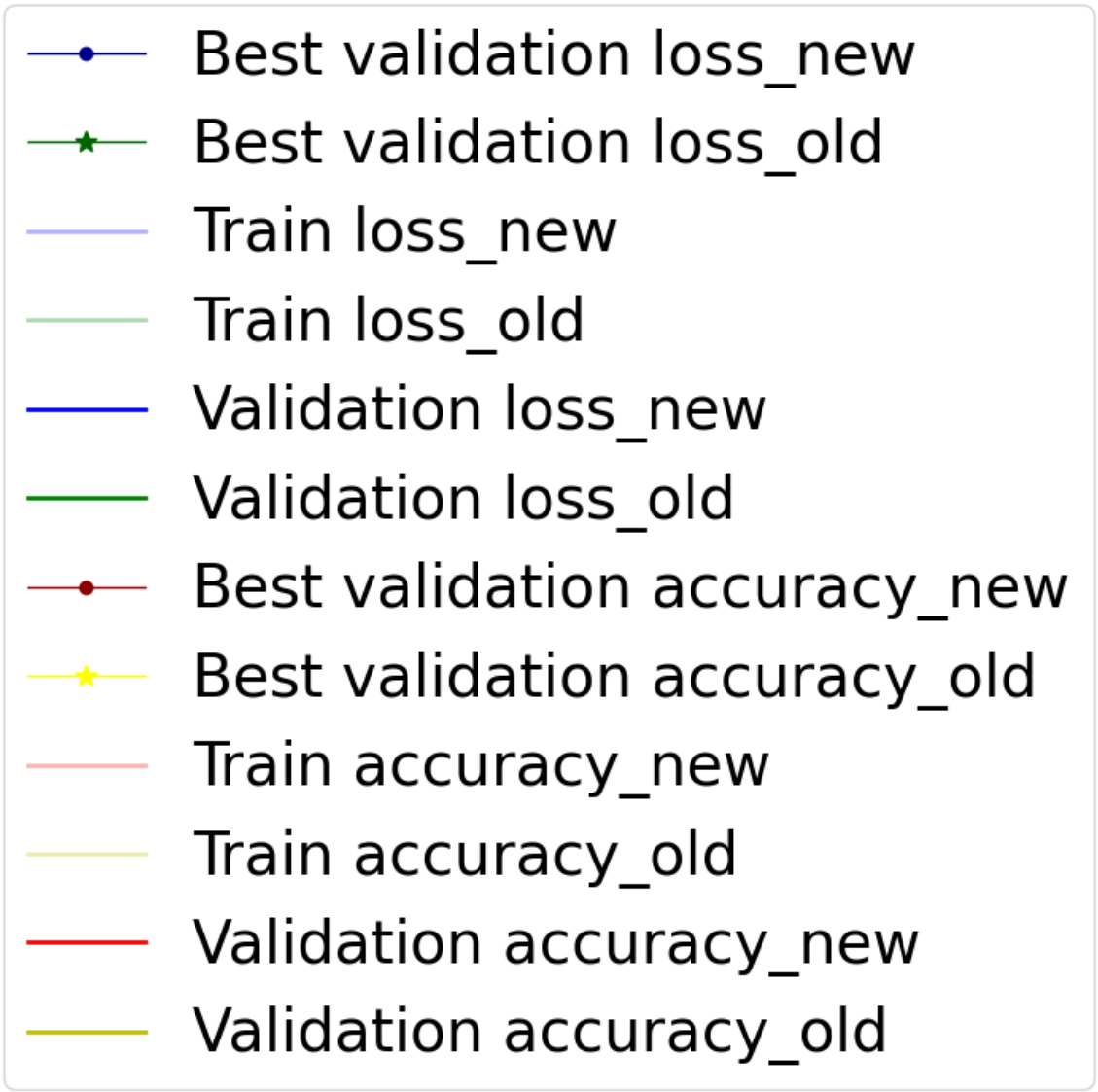
```
Help on function plot_training_progression in module analysis.classifica
tion:

plot_training_progression(self, plot_best=True, y_loss_lim=None, fig_siz
e=None, title=None, legend='center right')
    Plot the progression of training and validation loss and accuracy fr
om the run's logs

    Parameters
    ----------
    plot_best: bool, optional
        If true (default), plot points indicating the best validation lo
ss and accuracy
    y_loss_lim: (int, int), optional
        Range for the loss y-axis. By default, the range will expand to
show all loss values in the logs.
    fig_size: (float, float), optional
        Size of the figure
    title: str, optional
        Title of the figure. By default, do not plot a title.
    legend: str, optional
        Position to plot the legend. By default, the legend is placed in
the center right. For no legend use `None`.

    Returns
    -------
    matplotlib.figure.Figure
    matplotlib.axes.Axes
```

In [30]:
```
resnet_runs = pid_runs[:2] # first two are WatChMaL ResNet runs
runs = []
resnet_labels = ["ResNet (Basic cuts)", "ResNet (All cuts)"]
i = 0
for r1, r2 in resnet_runs:
    runs.append((r1.read_training_log_from_csv(directory[i]), r2.read_tra
```

```
    i += 2
i = 0
for a, b in runs:
    fig, ax1 = plt.subplots(figsize=(16,10))
    ax2 = ax1.twinx()
    ax1.set_title(resnet_labels[i])
    ax1.plot(a[3][a[-1]], a[4][a[-1]], lw=1, marker='o', label='Best vali
    ax1.plot(b[3][b[-1]], b[4][b[-1]], lw=1, marker='*', markersize = 10,
    ax1.plot(a[0], a[1], lw=2, label='Train loss_new', color='b', alpha=0
    ax1.plot(b[0], b[1], lw=2, label='Train loss_old', color='g', alpha=0
    ax1.plot(a[3], a[4], lw=2, label='Validation loss_new', color='b')
    ax1.plot(b[3], b[4], lw=2, label='Validation loss_old', color='g')
    ax2.plot(a[3][a[-1]], a[-2][a[-1]], lw=1, marker='o', label='Best val
    ax2.plot(b[3][b[-1]], b[-2][b[-1]] , lw=1, marker='*', markersize = 1
    ax2.plot(a[0], a[2], lw=2, label='Train accuracy_new', color='r', alp
    ax2.plot(b[0], b[2], lw=2, label='Train accuracy_old', color='y', alp
    ax2.plot(a[3], a[5], lw=2, label='Validation accuracy_new', color='r'
    ax2.plot(b[3], b[5], lw=2, label='Validation accuracy_old', color='y'
    ax2.set_ylabel("Accuracy", c='r')
    ax1.set_ylabel("Loss", c='b')
    ax1.set_xlabel("Epoch")
    i += 1
leg_fig, leg_ax = plot_legend((ax1, ax2))
```







Plot results

ROC curve for electron signal vs muon rejection

In [31]:
```
e_labels = [0, 1]
mu_labels = [2]
```

In [32]:
```
pid_runs = [
    WatChMaLClassification("/home/surajrai1900/WatChMaL/outputs/2023-09-1
    WatChMaLClassification("/home/pdeperio/machine_learning/data/IWCD_mPM
    WatChMaLClassification("/home/surajrai1900/WatChMaL/outputs/2023-09-1
    WatChMaLClassification("/home/pdeperio/machine_learning/data/IWCD_mPM
    FiTQunClassification(fq, "fiTQun (basic cuts)", h5_labels, h5_fq_offs
    FiTQunClassification(fq, "fiTQun (all cuts)", h5_labels, h5_fq_offset
]
```

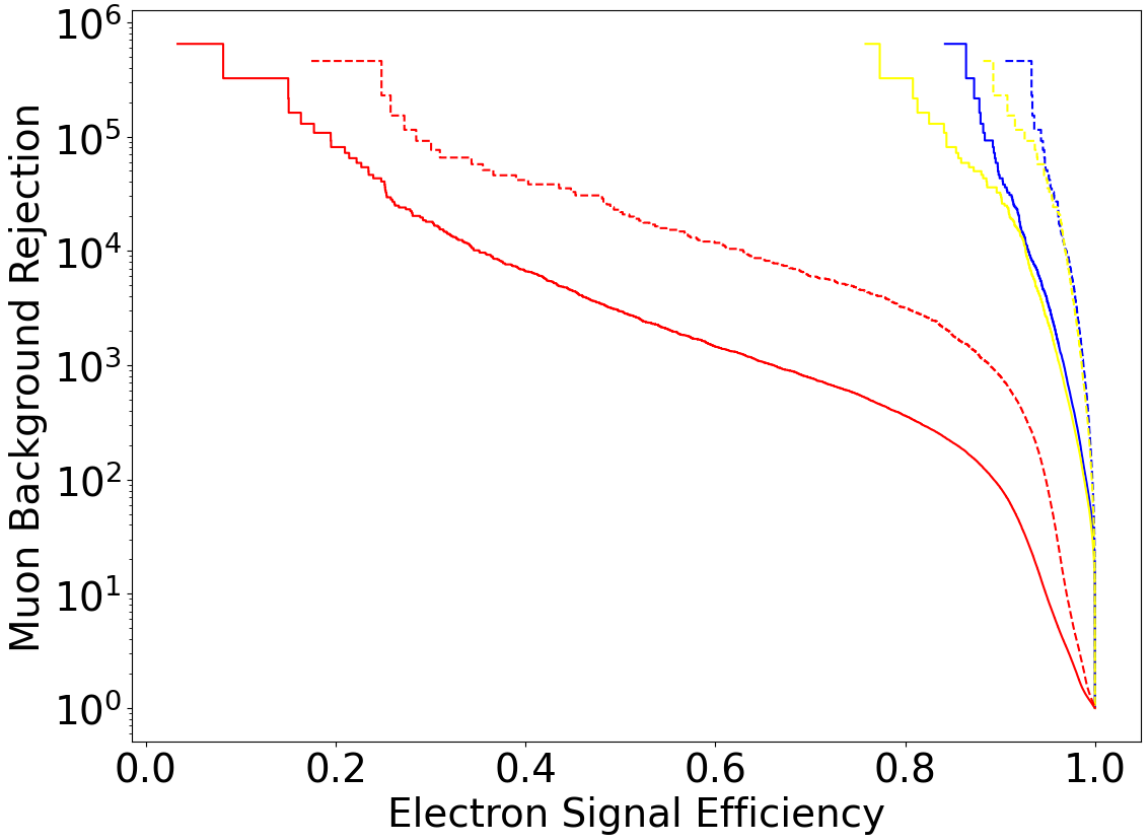
The `plot_rocs` function allows plotting overlaid ROC curves for a chosen signal and background classification.

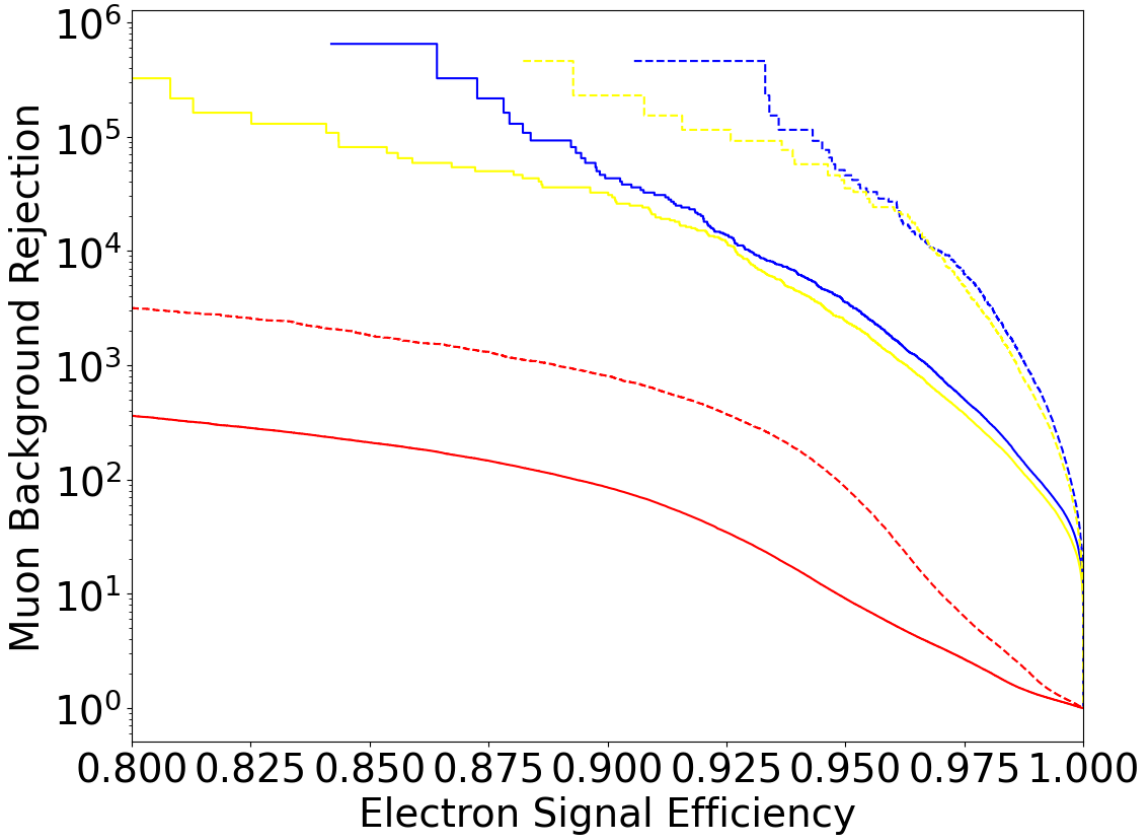
In [33]:
```
help(plot_rocs)
```

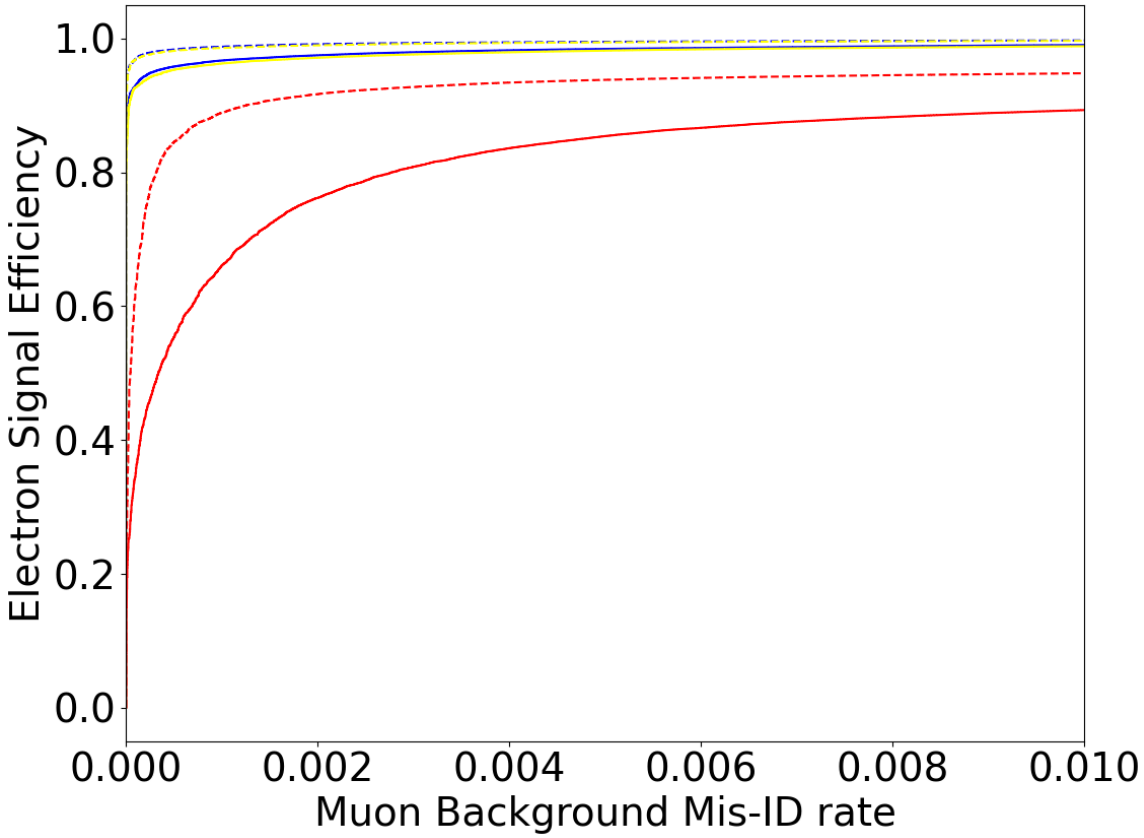
```
Help on function plot_rocs in module analysis.classification:

plot_rocs(runs, signal_labels, background_labels, selection=None, ax=Non
e, fig_size=None, x_label='', y_label='', x_lim=None, y_lim=None, y_log=
None, x_log=None, legend='best', mode='rejection', **plot_args)
    Plot overlaid ROC curves of results from a number of classification
runs

    Parameters
    ----------
    runs: sequence of ClassificationRun
        Sequence of runs to plot
    signal_labels: int or sequence of ints
        Set of labels corresponding to signal classes. Can be either a s
ingle label or a sequence of labels.
    background_labels: int or sequence of ints
        Set of labels corresponding to background classes. Can be either
a single label or a sequence of labels.
    selection: indexing expression, optional
        Selection of the discriminator values to be used (by default use
each run's predefined selection, or all events
        if none is defined).
    ax: matplotlib.axes.Axes
        Axes to draw the plot. If not provided, a new figure and axes is
created.
    fig_size: (float, float), optional
        Figure size. Ignored if `ax` is provided.
    x_label: str, optional
        Label of the x-axis.
    y_label: str, optional
        Label of the y-axis.
    x_lim: (float, float), optional
        Limits of the x-axis.
    y_lim: (float, float), optional
        Limits of the y-axis.
    x_log: bool, optional
        If True, plot the x-axis with log scale, otherwise use linear sc
ale (default).
    y_log: str, optional
        If True, plot the y-axis with log scale (default for 'rejection'
mode), otherwise use linear scale (default for
        'efficiency' mode).
    legend: str or None, optional
        Position of the legend, or None to have no legend. Attempts to f
ind the best position by default.
    mode: {'rejection', 'efficiency'}, optional
        If `rejection` (default) plot rejection factor (reciprocal of th
e false positive rate) on the y-axis versus
        signal efficiency (true positive rate) on the x-axis. If `effici
ency` plot background mis-ID rate (false
        positive rate) versus signal efficiency (true positive rate) on
the x-axis.
    plot_args: optional
        Additional arguments to pass to the `hist` plotting function. No
te that these may be overridden by arguments
        defined in `runs`.

    Returns
    -------
    fig: matplotlib.figure.Figure
```

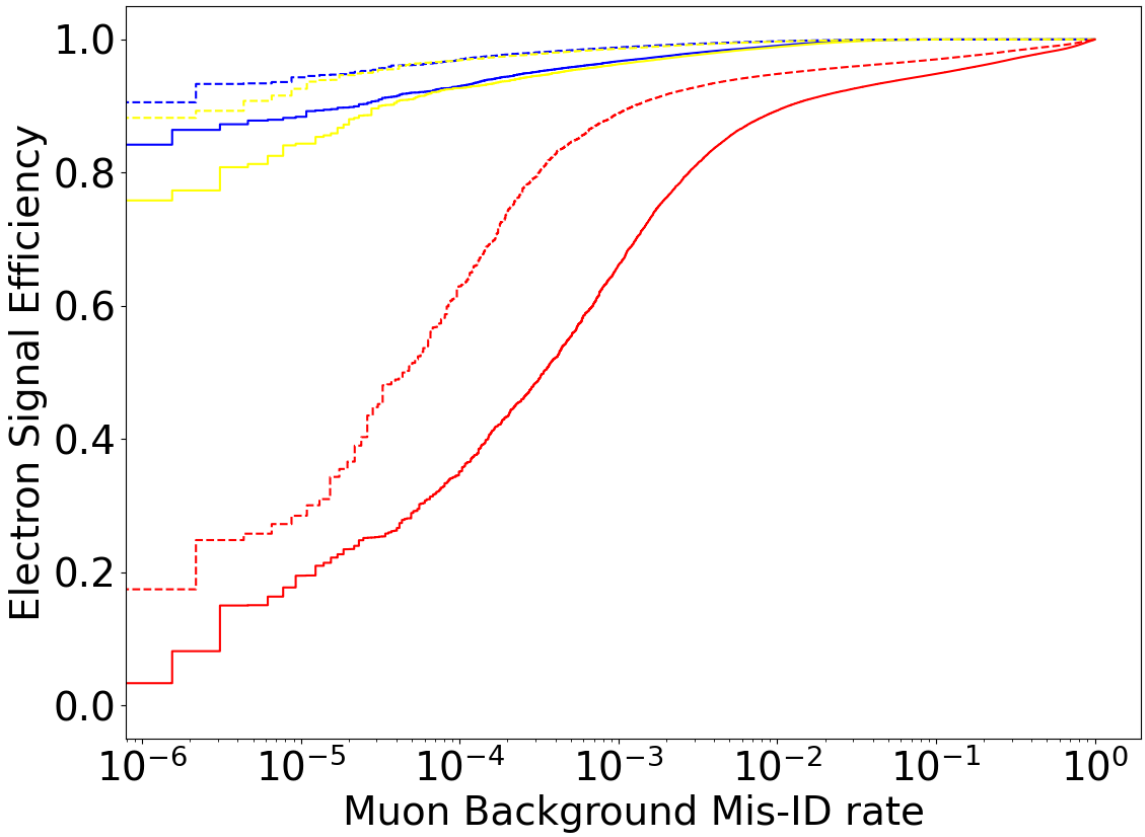
```
ax: matplotlib.axes.Axes
```

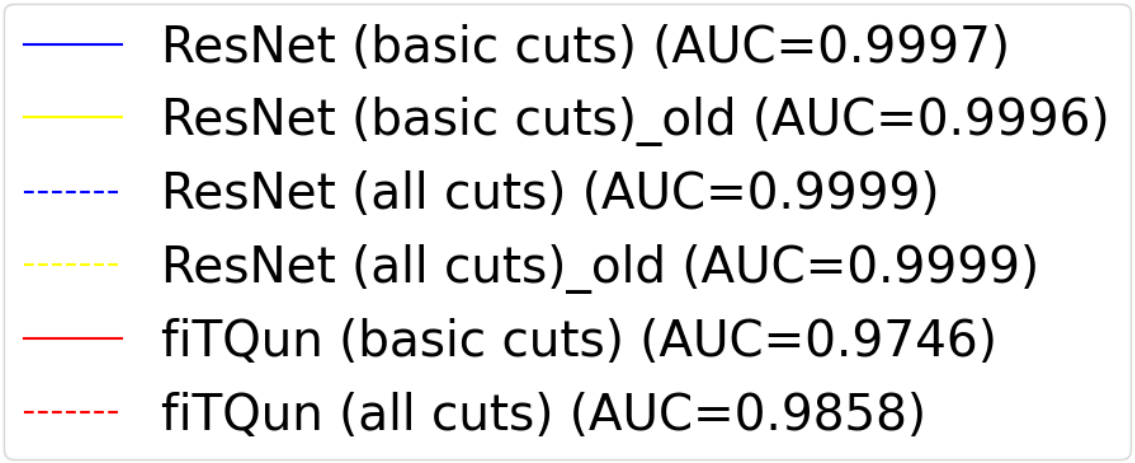
In [34]:
```
fig, ax = plot_rocs(pid_runs, e_labels, mu_labels, x_label="Electron Sign
fig, ax = plot_rocs(pid_runs, e_labels, mu_labels, x_label="Electron Sign
fig, ax = plot_rocs(pid_runs, e_labels, mu_labels, x_label="Muon Backgrou
fig, ax = plot_rocs(pid_runs, e_labels, mu_labels, x_label="Muon Backgrou
l_fig, l_ax = plot_legend(ax)
```






Electron Signal Efficiency
1.0
0.8
0.6
0.4
0.2
0.0
0.000
0.002
0.004
0.006
0.008
0.010
Muon Background Mis-ID rate


Electron Signal Efficiency
1.0
0.8
0.6
0.4
0.2
0.0
10^{-6}
10^{-5}
10^{-4}
10^{-3}
10^{-2}
10^{-1}
10^{0}
Muon Background Mis-ID rate



for fixed 0.1% muon mis-ID

for each 50 MeV bin of reconstructed momentum, calculate the thresholds that reject 99.9% of muons and apply cut to all events

In [35]:
```
help(WatChMaLClassification.cut_with_fixed_efficiency)
```

```
Help on function cut_with_fixed_efficiency in module analysis.classifica
tion:

cut_with_fixed_efficiency(self, signal_labels, background_labels, effici
ency, selection=None, select_labels=None, return_threshold=False)
    Generate array of boolean values indicating whether each event passe
s a cut defined such that a fixed proportion
    of the selected events pass the cut.
    After taking the subset of `discriminator_values` defined by `select
ion`, the threshold discriminator value is
    found such that the proportion that are above the threshold is equal
to `efficiency`. This cut threshold is then
    used to apply the cut to all events (not just those selected by `sel
ection`) and an array of booleans is
    returned for whether each discriminator value is above the threshold
of its corresponding bin. The cut result is
    also stored for use in plotting efficiency profiles.

    Parameters
    ----------
    signal_labels: int or sequence of ints
        Set of labels corresponding to signal classes. Can be either a s
ingle label or a sequence of labels.
    background_labels: int or sequence of ints
        Set of labels corresponding to background classes. Can be either
a single label or a sequence of labels.
    efficiency: float
        The fixed efficiency.
    selection: indexing expression, optional
        Selection of the discriminator values to use in calculating the
threshold applied by the cut (by default use
        all values).
    select_labels: set of int
        Set of true labels to select events to use.
    return_threshold: bool, optional
        If True, return also the cut threshold.

    Returns
    -------
    cut: np.ndarray of bool
        One-dimensional array the same length as `discriminator_values`
indicating whether each event passes the cut.
    threshold: float, optional
        The threshold applied by the cut.
```

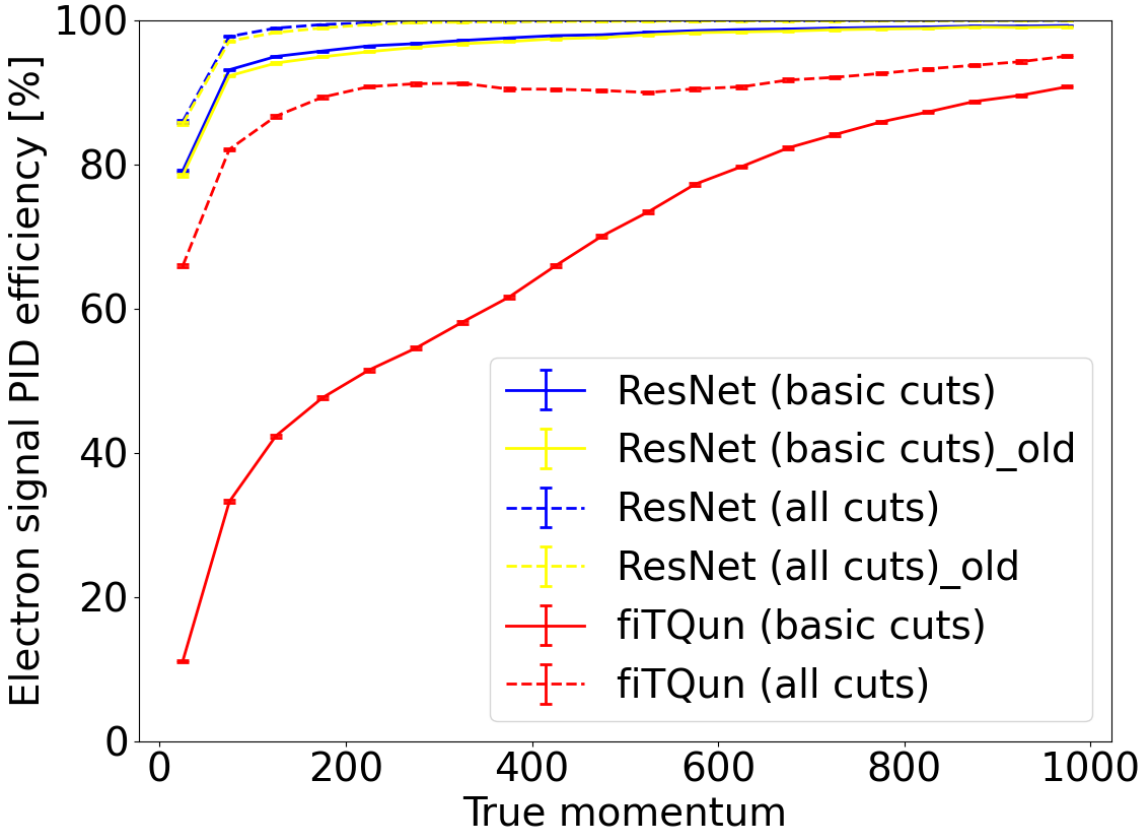
In [36]:
```
muon_rejection = 0.999
muon_efficiency = 1-muon_rejection
for r in pid_runs:
    r.cut_with_fixed_efficiency(e_labels, mu_labels, muon_efficiency, sel
```

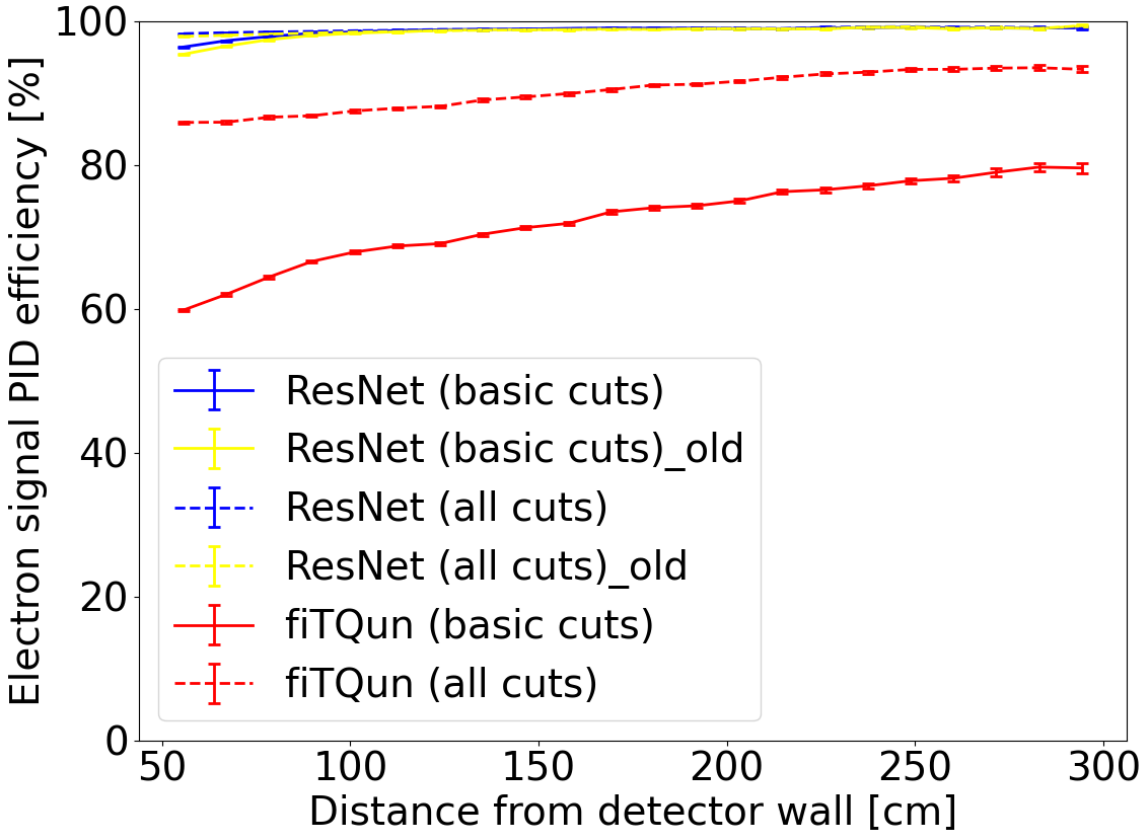
Plot signal efficiency against true momentum, dwall, towall, zenith, azimuth

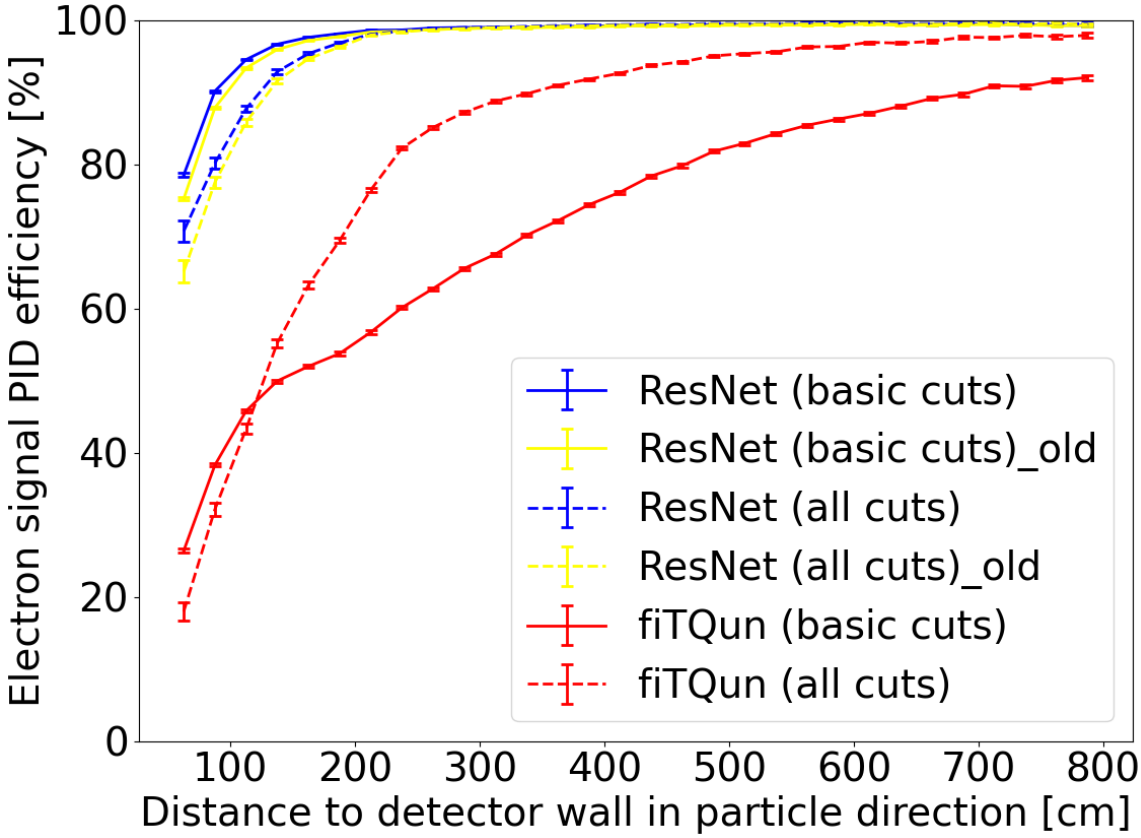
In [37]:
```
help(plot_efficiency_profile)
```

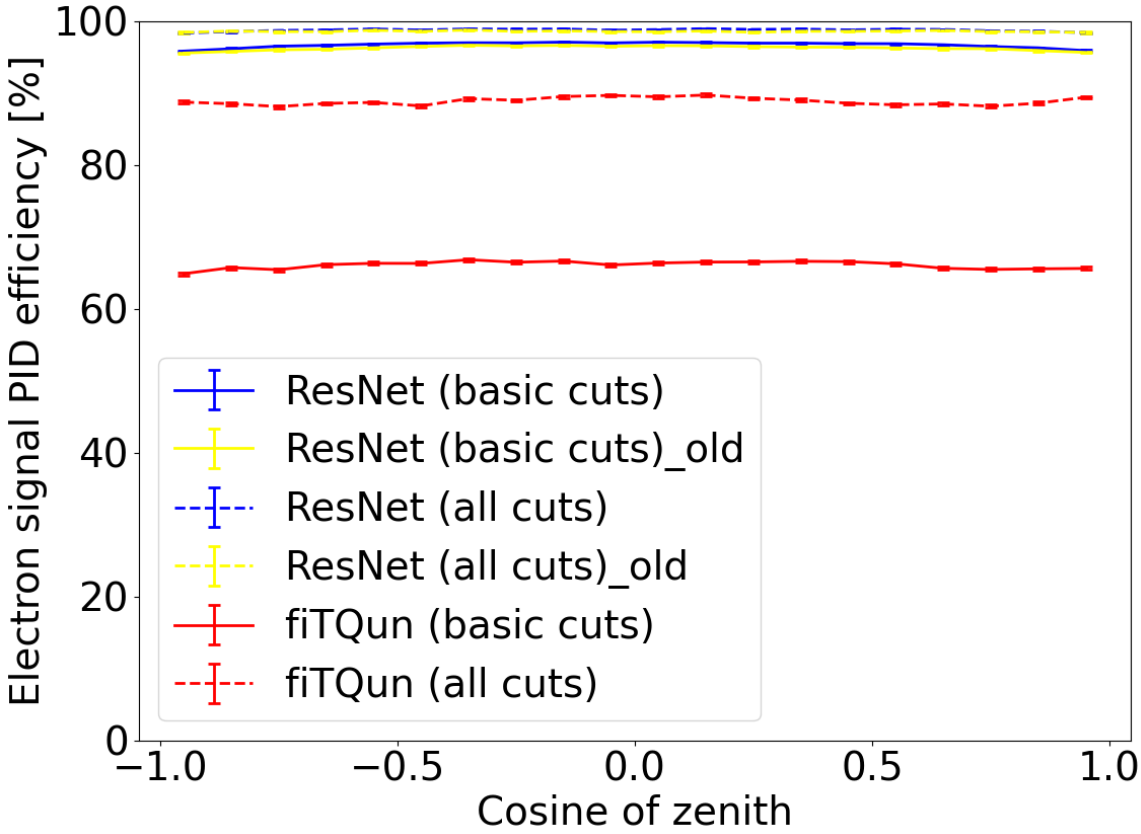
```
Help on function plot_efficiency_profile in module analysis.classificati
on:

plot_efficiency_profile(runs, binning, selection=None, select_labels=Non
e, ax=None, fig_size=None, x_label='', y_label='', legend='best', y_lim=
None, **plot_args)
    Plot binned efficiencies for a cut applied to a number of classifica
tion runs.
    Each run should already have had a cut generated, then in each bin t
he proportion of events passing the cut is
    calculated as the efficiency and plotted. A selection can be provide
d to use only a subset of all the values. The
    same binning and selection is applied to each run.

    Parameters
    ----------
    runs: sequence of ClassificationRun
        Sequence of runs to plot
    binning: (np.ndarray, np.ndarray)
        Array of bin edges and array of bin indices, returned from `anal
ysis.utils.binning.get_binning`.
    selection: indexing expression, optional
        Selection of the values to use in calculating the efficiencies
(by default use each run's predefined selection,
        or all events if none is defined).
    select_labels: set of int, optional
        Set of true labels to select events to use.
    ax: matplotlib.axes.Axes
        Axes to draw the plot. If not provided, a new figure and axes is
created.
    fig_size: (float, float), optional
        Figure size. Ignored if `ax` is provided.
    x_label: str, optional
        Label of the x-axis.
    y_label: str, optional
        Label of the y-axis.
    legend: str or None, optional
        Position of the legend, or None to have no legend. Attempts to f
ind the best position by default.
    y_lim: (float, float), optional
        Limits of the y-axis.
    plot_args: optional
        Additional arguments to pass to the plotting function. Note that
these may be overridden by arguments
        provided in `runs`.

    Returns
    -------
    fig: matplotlib.figure.Figure
    ax: matplotlib.axes.Axes
```

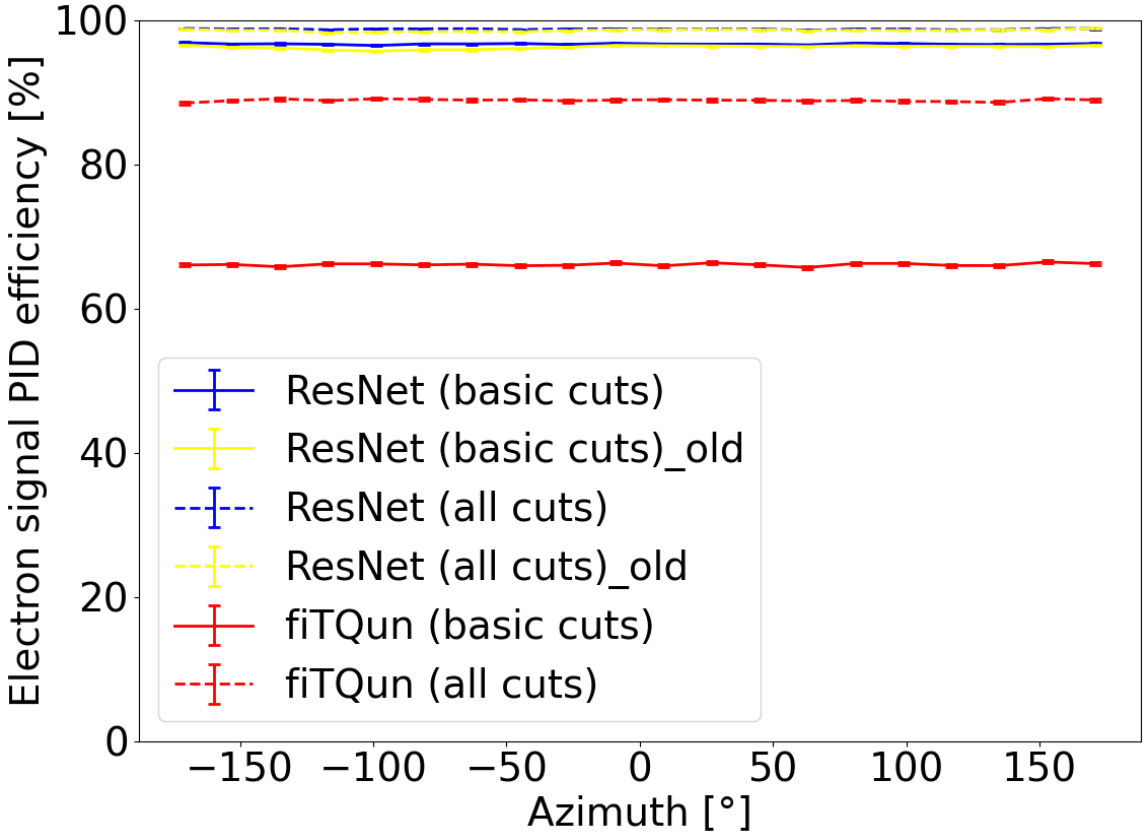
In [38]:
```
fig, ax = plot_efficiency_profile(pid_runs, mom_binning, select_labels=e_
fig, ax = plot_efficiency_profile(pid_runs, dwall_binning, select_labels=
fig, ax = plot_efficiency_profile(pid_runs, towall_binning, select_labels
fig, ax = plot_efficiency_profile(pid_runs, cos_zenith_binning, select_la
fig, ax = plot_efficiency_profile(pid_runs, azimuth_binning, select_label
```


Electron signal PID efficiency [%]
True momentum
ResNet (basic cuts)
ResNet (basic cuts)_old
ResNet (all cuts)
ResNet (all cuts)_old
fiTQun (basic cuts)
fiTQun (all cuts)


Electron signal PID efficiency [%]
Distance from detector wall [cm]
ResNet (basic cuts)
ResNet (basic cuts)_old
ResNet (all cuts)
ResNet (all cuts)_old
fiTQun (basic cuts)
fiTQun (all cuts)

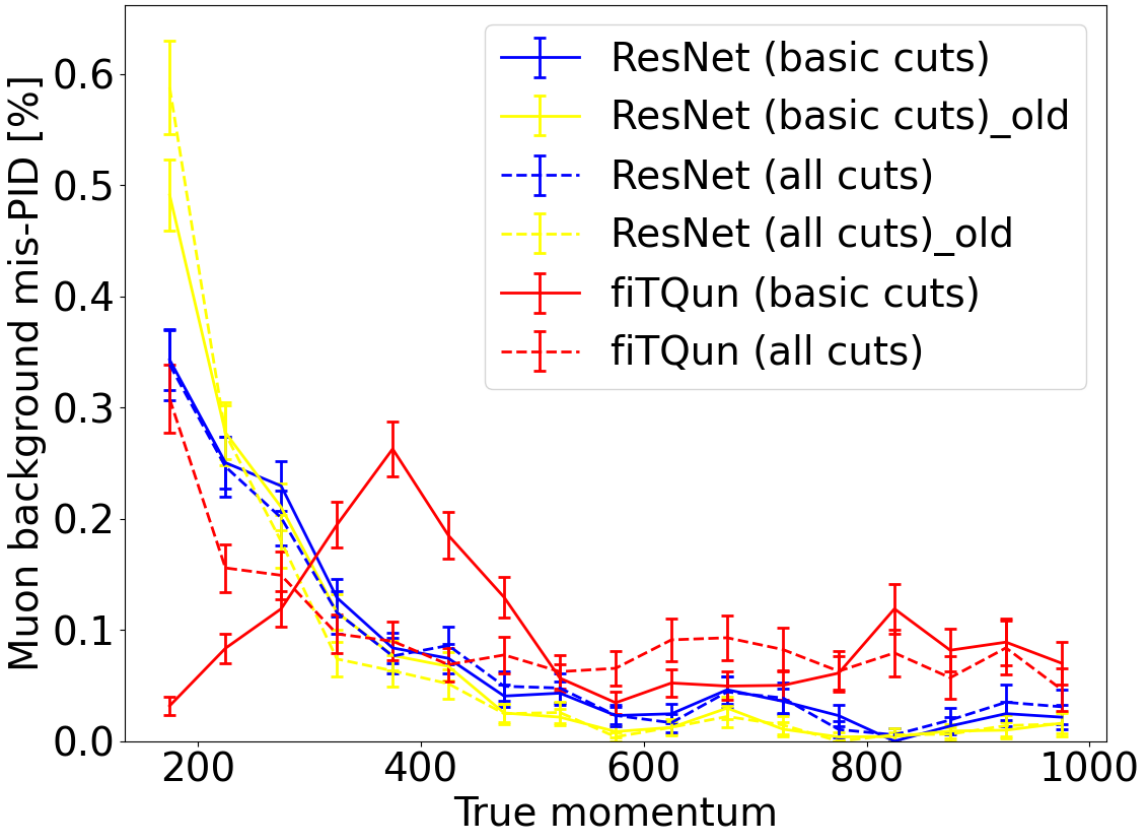

Electron signal PID efficiency [%]
Distance to detector wall in particle direction [cm]
ResNet (basic cuts)
ResNet (basic cuts)_old
ResNet (all cuts)
ResNet (all cuts)_old
fiTQun (basic cuts)
fiTQun (all cuts)

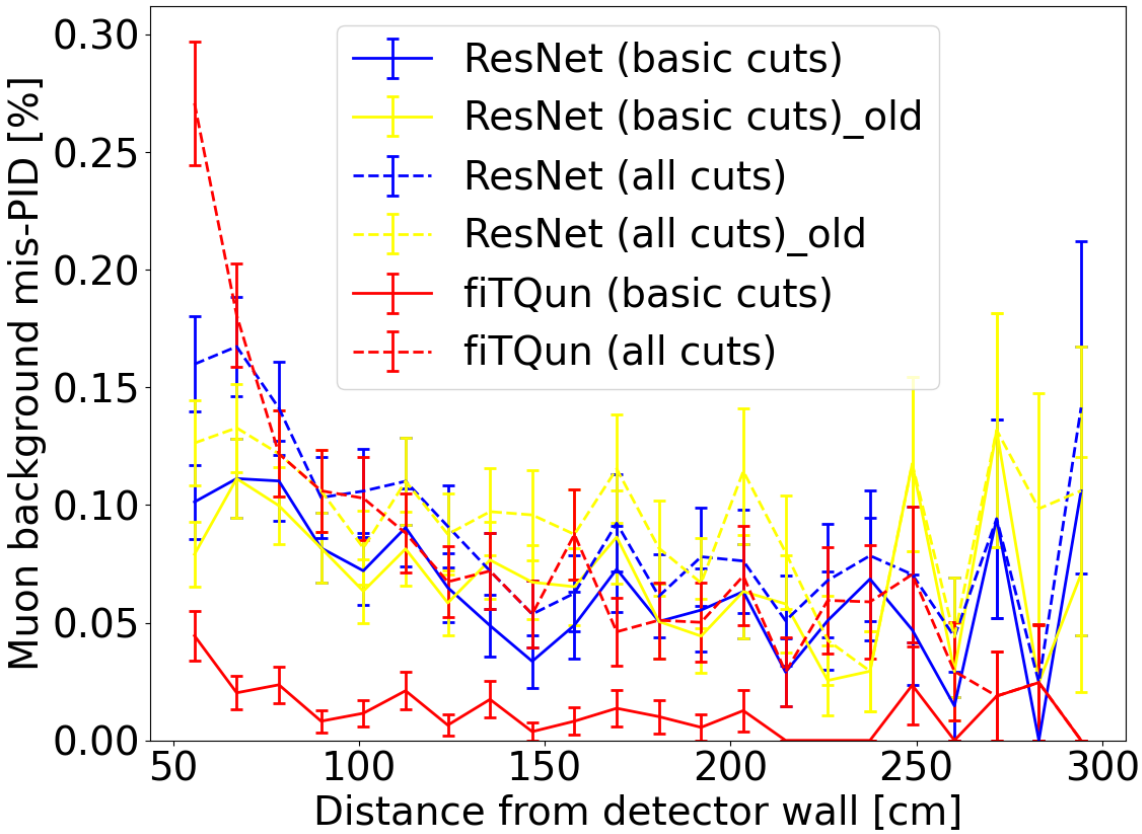

Electron signal PID efficiency [%]
Cosine of zenith
ResNet (basic cuts)
ResNet (basic cuts)_old
ResNet (all cuts)
ResNet (all cuts)_old
fiTQun (basic cuts)
fiTQun (all cuts)

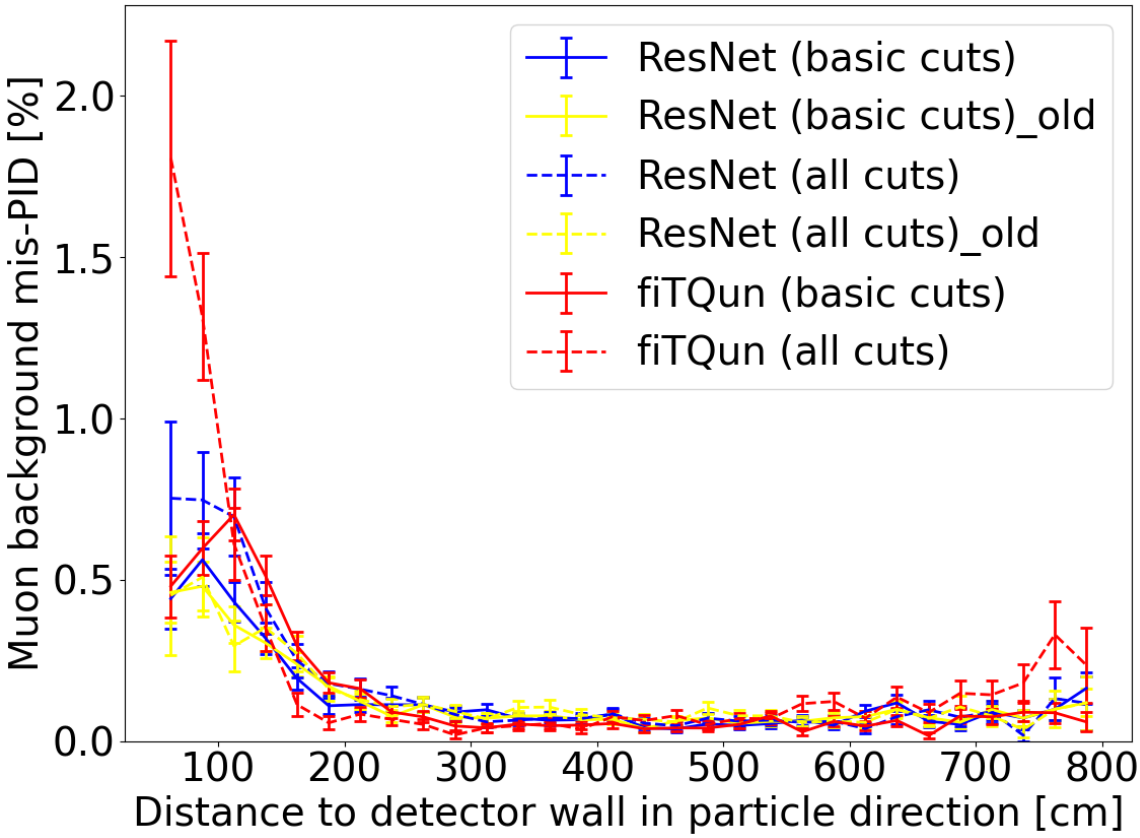


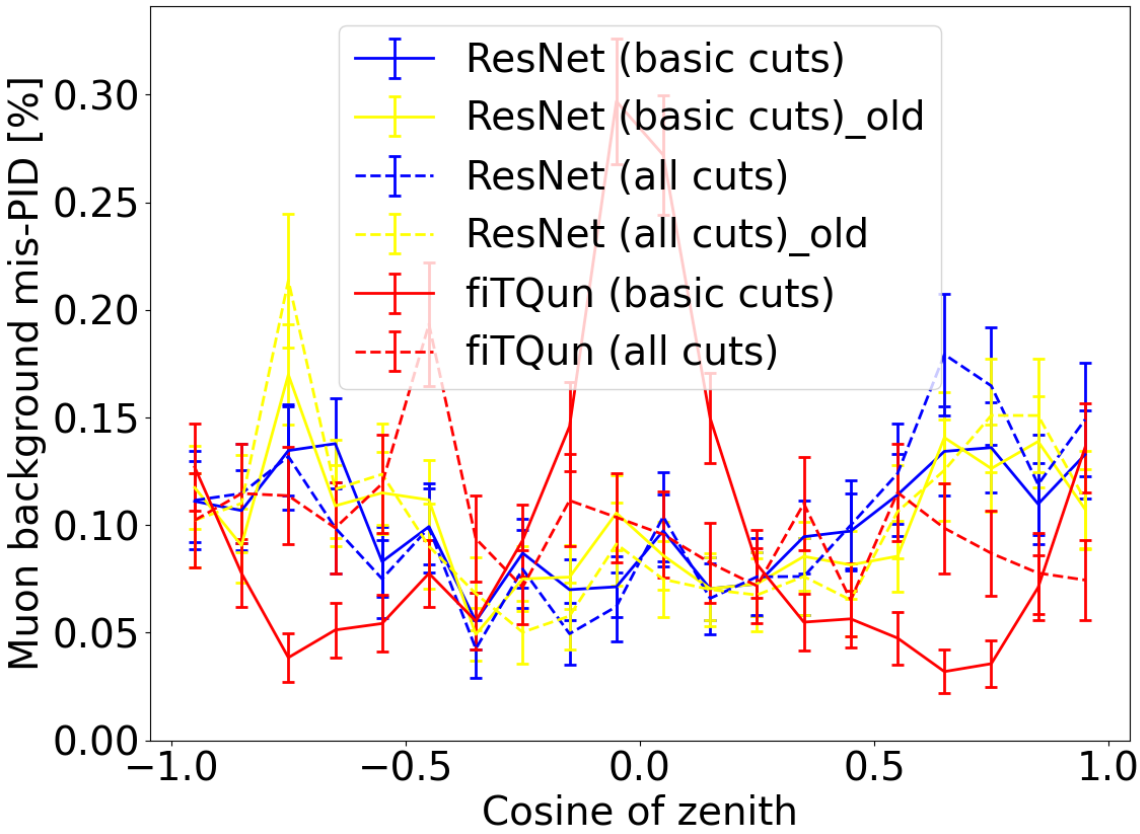
Plot background mis-ID against true momentum, dwall, towall, zenith, azimuth

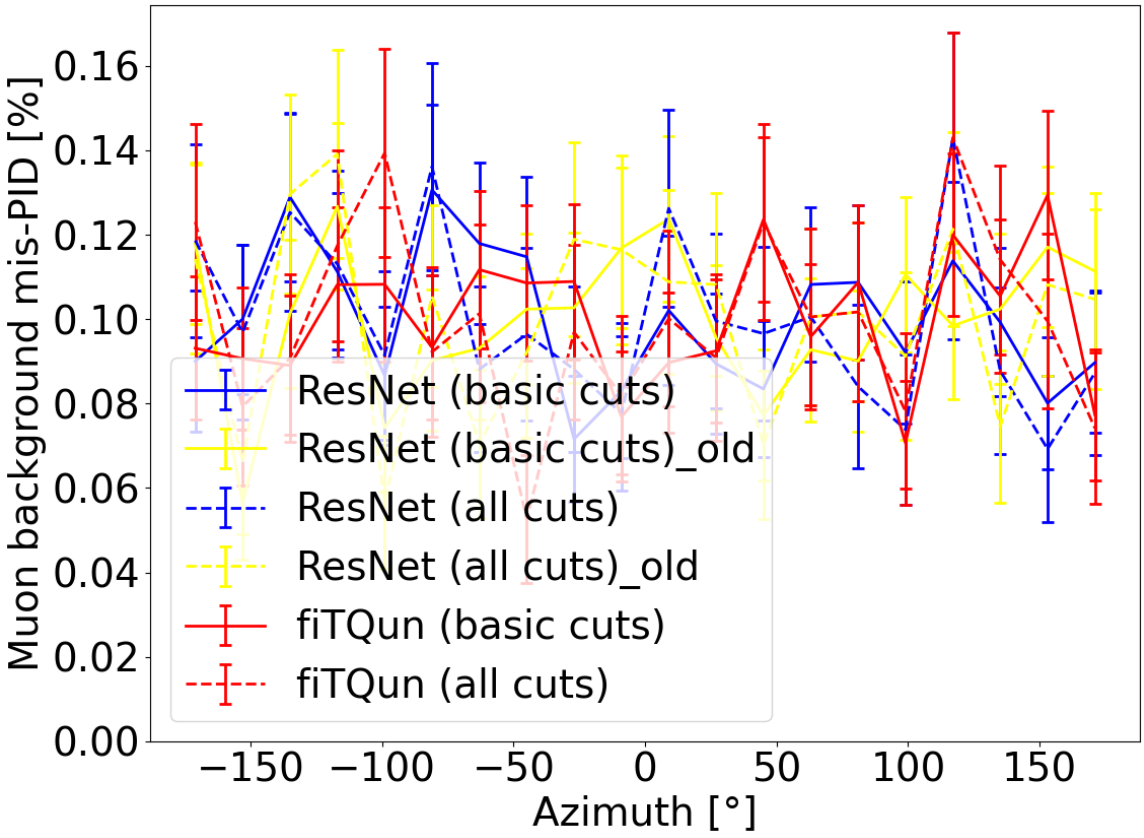
In [39]:
```
fig, ax = plot_efficiency_profile(pid_runs, mom_binning_mu, select_labels
fig, ax = plot_efficiency_profile(pid_runs, dwall_binning, select_labels=
fig, ax = plot_efficiency_profile(pid_runs, towall_binning, select_labels
fig, ax = plot_efficiency_profile(pid_runs, cos_zenith_binning, select_la
fig, ax = plot_efficiency_profile(pid_runs, azimuth_binning, select_label
```


ResNet (basic cuts)
ResNet (basic cuts)_old
ResNet (all cuts)
ResNet (all cuts)_old
fiTQun (basic cuts)
fiTQun (all cuts)
Muon background mis-PID [%]
0.0
0.1
0.2
0.3
0.4
0.5
0.6
200
400
600
800
1000
True momentum


ResNet (basic cuts)
ResNet (basic cuts)_old
ResNet (all cuts)
ResNet (all cuts)_old
fiTQun (basic cuts)
fiTQun (all cuts)
Muon background mis-PID [%]
0.00
0.05
0.10
0.15
0.20
0.25
0.30
50
100
150
200
250
300
Distance from detector wall [cm]


ResNet (basic cuts)
ResNet (basic cuts)_old
ResNet (all cuts)
ResNet (all cuts)_old
fiTQun (basic cuts)
fiTQun (all cuts)
Muon background mis-PID [%]
0.0
0.5
1.0
1.5
2.0
100
200
300
400
500
600
700
800
Distance to detector wall in particle direction [cm]


ResNet (basic cuts)
ResNet (basic cuts)_old
ResNet (all cuts)
ResNet (all cuts)_old
fiTQun (basic cuts)
fiTQun (all cuts)
Muon background mis-PID [%]
0.00
0.05
0.10
0.15
0.20
0.25
0.30
−1.0
−0.5
0.0
0.5
1.0
Cosine of zenith



Electron vs gamma

Load results

Load ResNet and fiTQun results

In [40]:
```
pid_runs = [
    WatChMaLClassification("/home/surajrai1900/WatChMaL/outputs/2023-09-1
    WatChMaLClassification("/home/pdeperio/machine_learning/data/IWCD_mPM
    WatChMaLClassification("/home/surajrai1900/WatChMaL/outputs/2023-09-1
    WatChMaLClassification("/home/pdeperio/machine_learning/data/IWCD_mPM
    FiTQunClassification(fq, "fiTQun (basic cuts)", h5_labels, h5_fq_offs
    FiTQunClassification(fq, "fiTQun (all cuts)", h5_labels, h5_fq_offset
]
```

Plot results

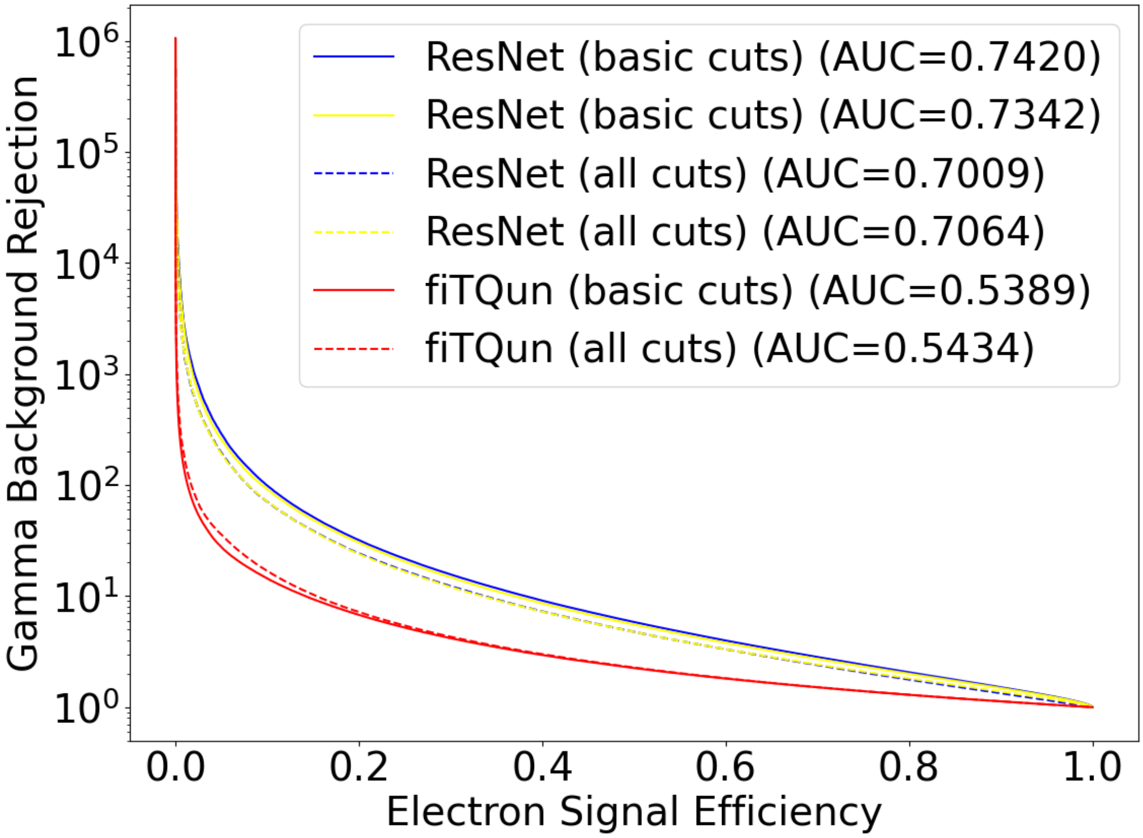
ROC curve for electron signal vs gamma rejection

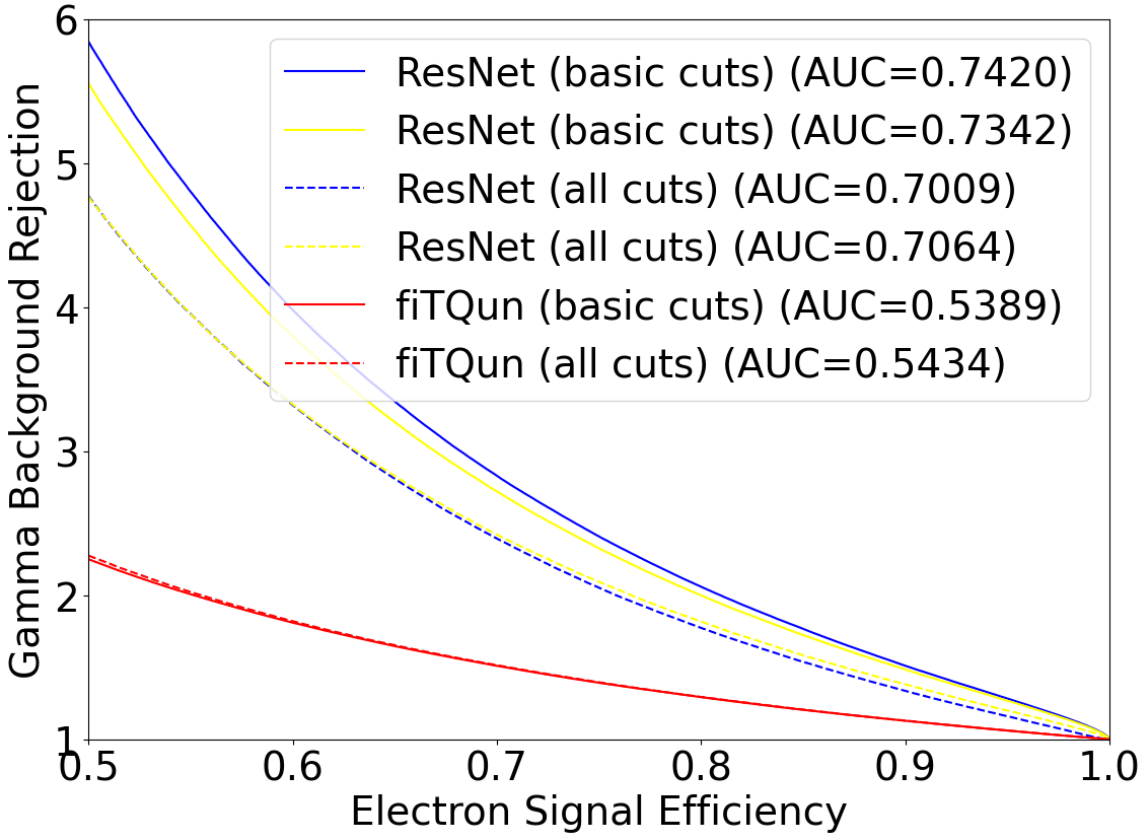
In [41]:
```
e_label = [1]
g_label = [0]
```

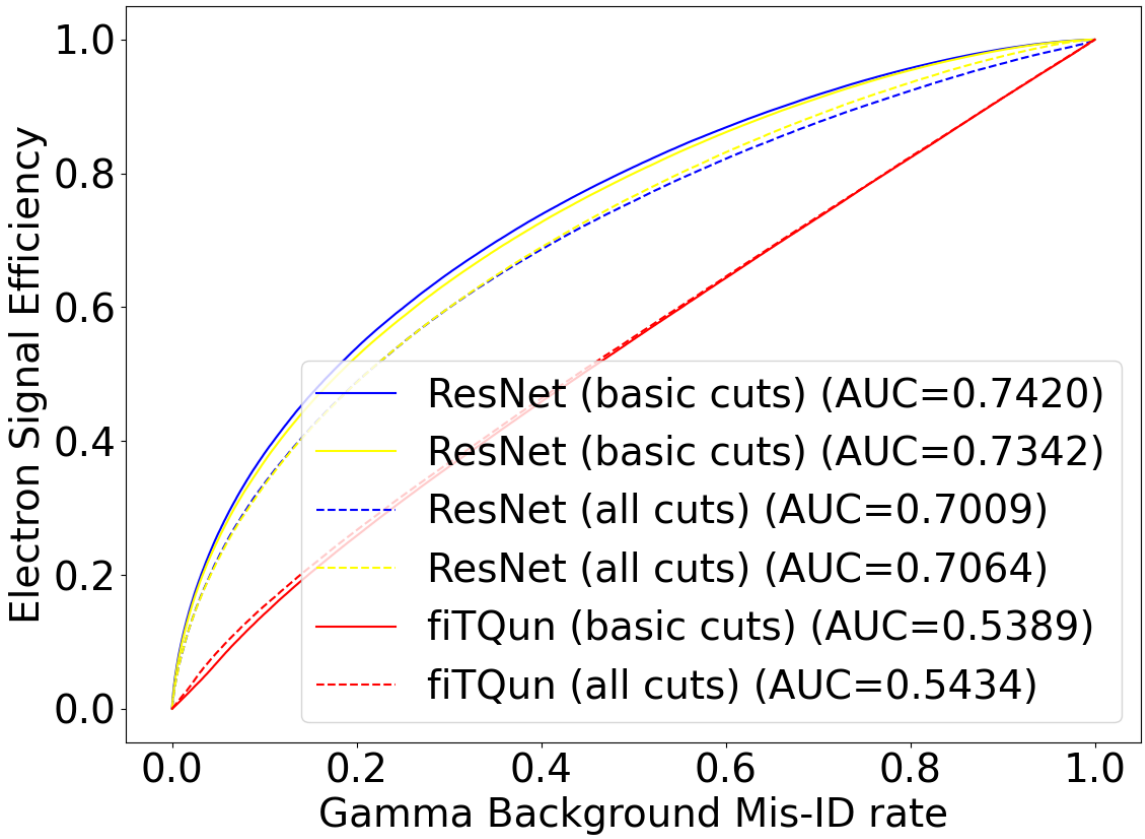
In [42]:
```
fig, ax = plot_rocs(pid_runs, e_label, g_label, x_label="Electron Signal
fig, ax = plot_rocs(pid_runs, e_label, g_label, x_label="Electron Signal
fig, ax = plot_rocs(pid_runs, e_label, g_label, x_label="Gamma Background
```

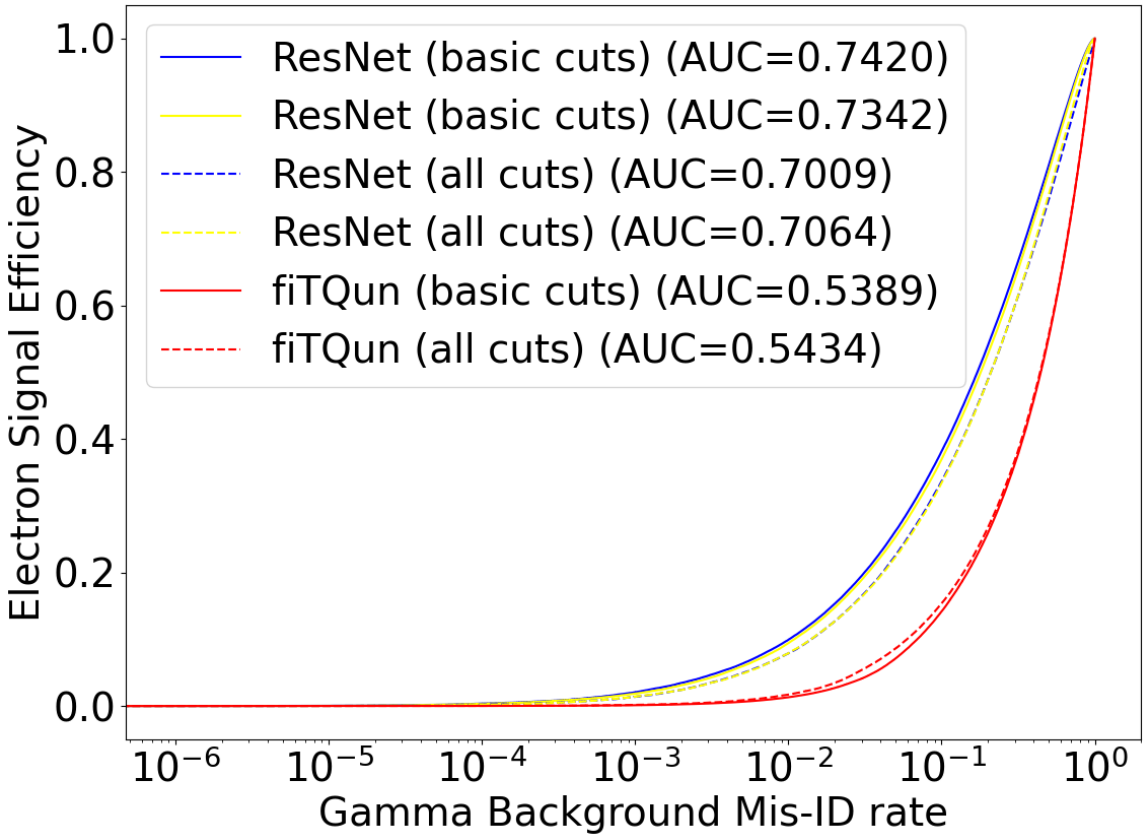
```
fig, ax = plot_rocs(pid_runs, e_label, g_label, x_label="Gamma Backgrounc
l_fig, l_ax = plot_legend(ax)
```

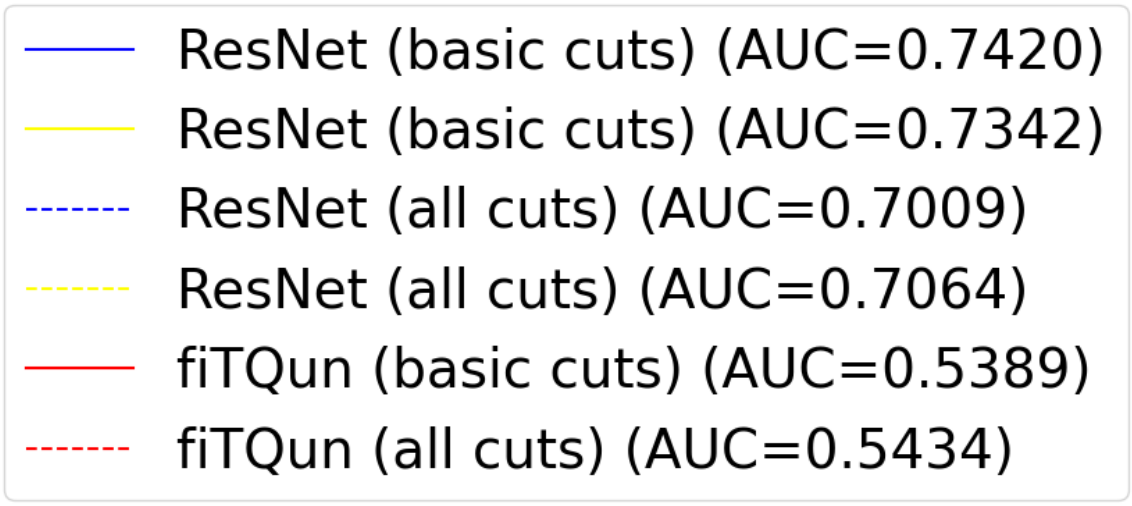
```
/home/surajrai1900/WatChMaL/analysis/classification.py:497: RuntimeWarni
ng: invalid value encountered in true_divide
  return signal_softmax / (signal_softmax + background_softmax)
/home/surajrai1900/WatChMaL/analysis/classification.py:497: RuntimeWarni
ng: invalid value encountered in true_divide
  return signal_softmax / (signal_softmax + background_softmax)
/home/surajrai1900/WatChMaL/analysis/classification.py:497: RuntimeWarni
ng: invalid value encountered in true_divide
  return signal_softmax / (signal_softmax + background_softmax)
/home/surajrai1900/WatChMaL/analysis/classification.py:497: RuntimeWarni
ng: invalid value encountered in true_divide
  return signal_softmax / (signal_softmax + background_softmax)
/home/surajrai1900/WatChMaL/analysis/classification.py:497: RuntimeWarni
ng: invalid value encountered in true_divide
  return signal_softmax / (signal_softmax + background_softmax)
/home/surajrai1900/WatChMaL/analysis/classification.py:497: RuntimeWarni
ng: invalid value encountered in true_divide
  return signal_softmax / (signal_softmax + background_softmax)
/home/surajrai1900/WatChMaL/analysis/classification.py:497: RuntimeWarni
ng: invalid value encountered in true_divide
  return signal_softmax / (signal_softmax + background_softmax)
/home/surajrai1900/WatChMaL/analysis/classification.py:497: RuntimeWarni
ng: invalid value encountered in true_divide
  return signal_softmax / (signal_softmax + background_softmax)
```




ResNet (basic cuts) (AUC=0.7420)
ResNet (basic cuts) (AUC=0.7342)
ResNet (all cuts) (AUC=0.7009)
ResNet (all cuts) (AUC=0.7064)
fiTQun (basic cuts) (AUC=0.5389)
fiTQun (all cuts) (AUC=0.5434)
Gamma Background Rejection
Electron Signal Efficiency
6
5
4
3
2
1
0.5
0.6
0.7
0.8
0.9
1.0


ResNet (basic cuts) (AUC=0.7420)
ResNet (basic cuts) (AUC=0.7342)
ResNet (all cuts) (AUC=0.7009)
ResNet (all cuts) (AUC=0.7064)
fiTQun (basic cuts) (AUC=0.5389)
fiTQun (all cuts) (AUC=0.5434)
Electron Signal Efficiency
Gamma Background Mis-ID rate
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

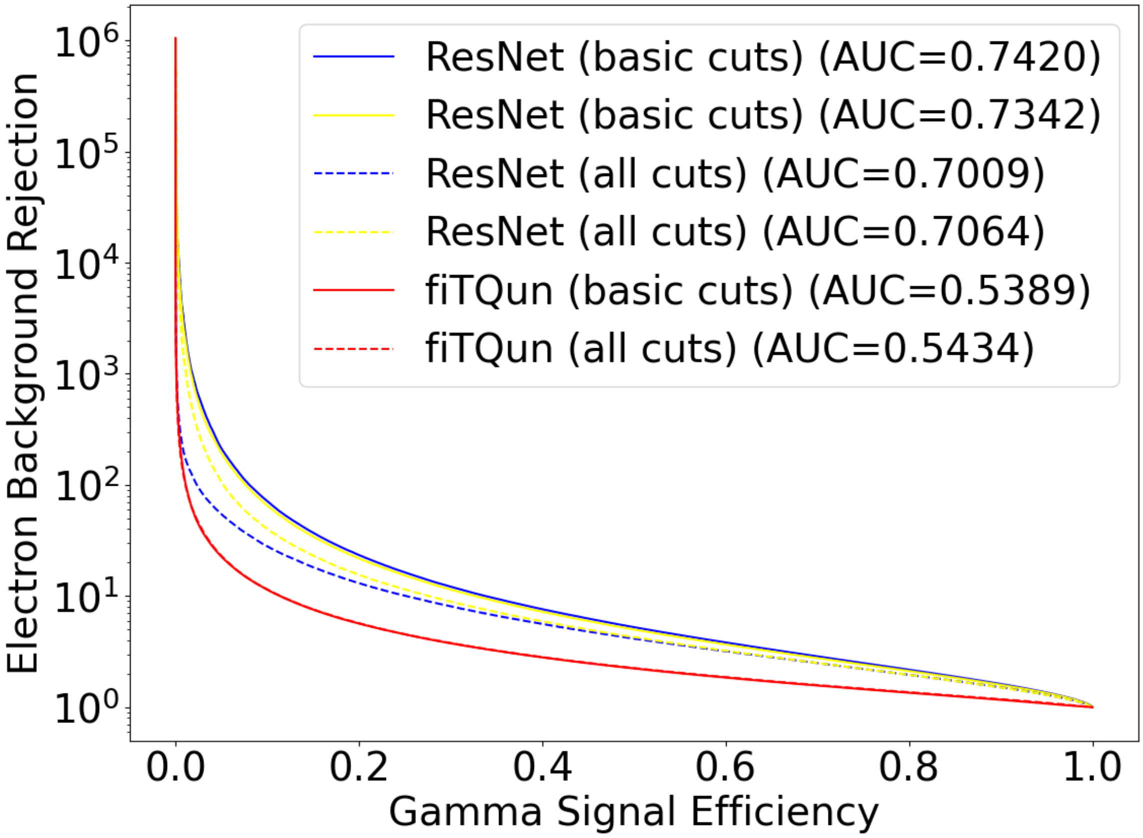


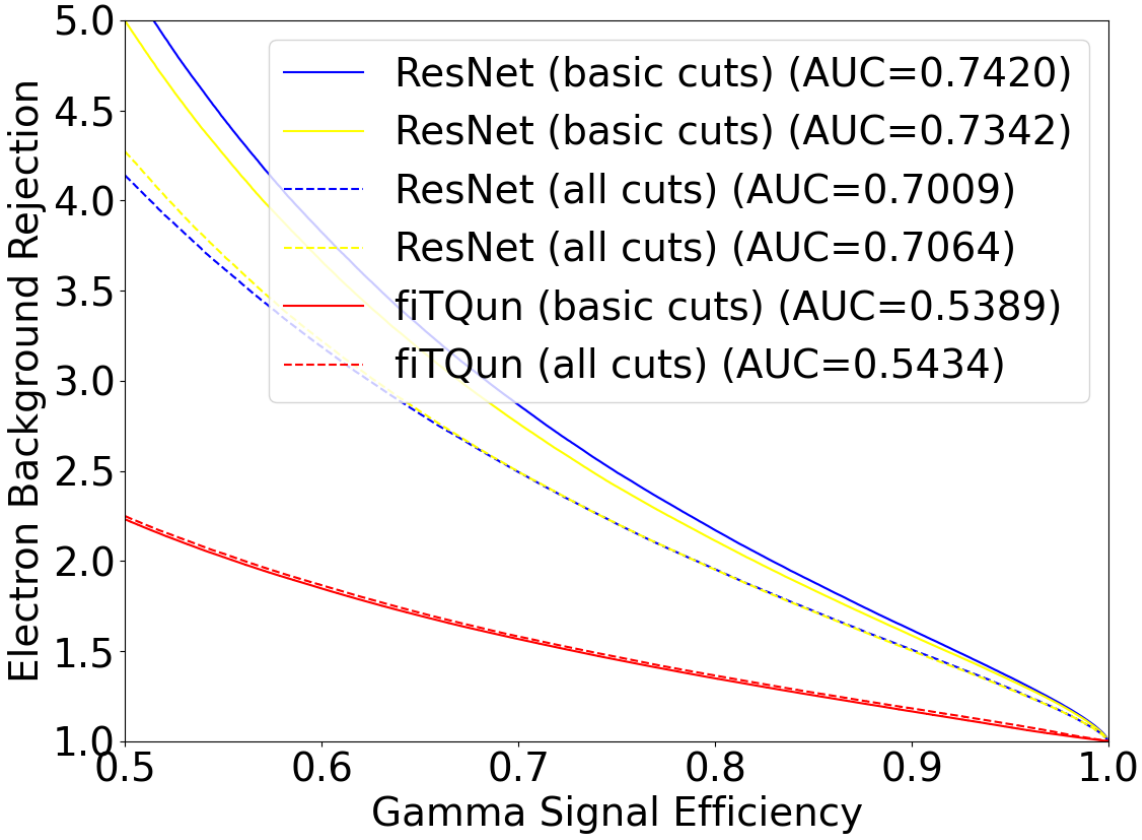


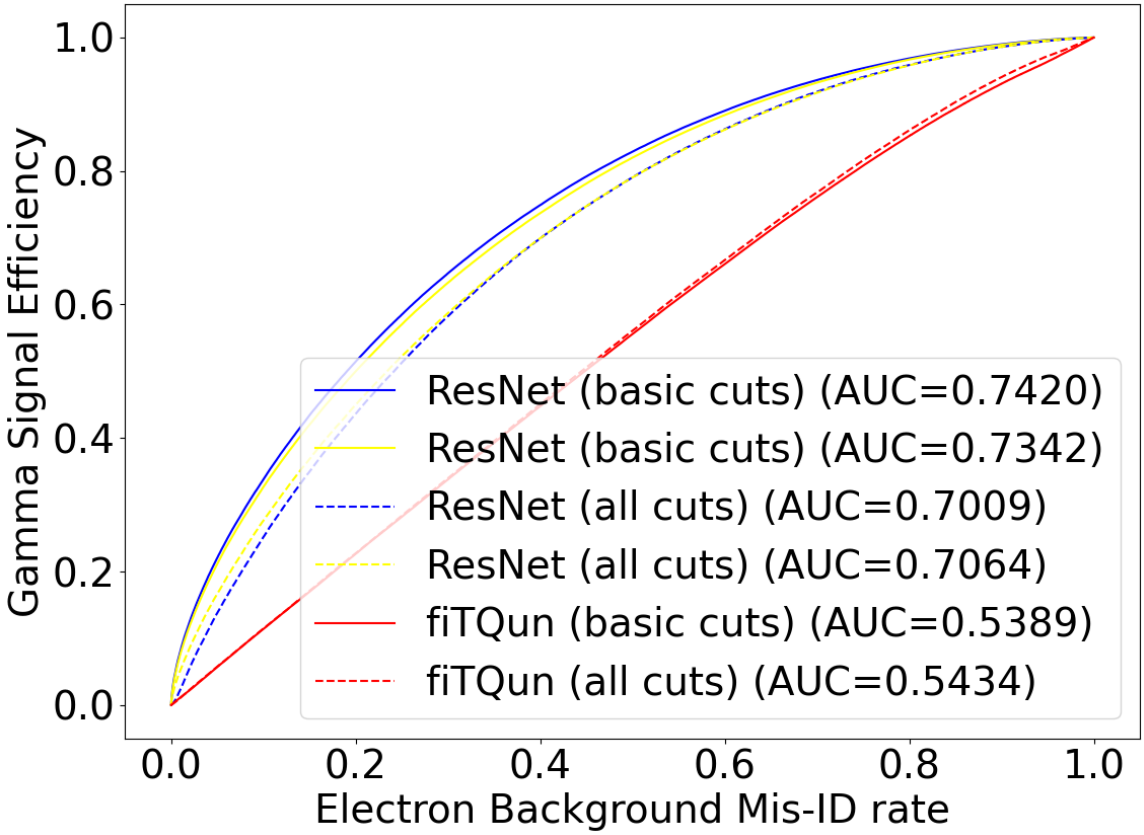
ROC curve for gamma signal vs electron rejection

In [43]:
```
fig, ax = plot_rocs(pid_runs, g_label, e_label, x_label="Gamma Signal Eff
fig, ax = plot_rocs(pid_runs, g_label, e_label, x_label="Gamma Signal Eff
fig, ax = plot_rocs(pid_runs, g_label, e_label, x_label="Electron Backgro
```

```
/home/surajrai1900/WatChMaL/analysis/classification.py:497: RuntimeWarni
ng: invalid value encountered in true_divide
  return signal_softmax / (signal_softmax + background_softmax)
/home/surajrai1900/WatChMaL/analysis/classification.py:497: RuntimeWarni
ng: invalid value encountered in true_divide
  return signal_softmax / (signal_softmax + background_softmax)
/home/surajrai1900/WatChMaL/analysis/classification.py:497: RuntimeWarni
ng: invalid value encountered in true_divide
  return signal_softmax / (signal_softmax + background_softmax)
/home/surajrai1900/WatChMaL/analysis/classification.py:497: RuntimeWarni
ng: invalid value encountered in true_divide
  return signal_softmax / (signal_softmax + background_softmax)
/home/surajrai1900/WatChMaL/analysis/classification.py:497: RuntimeWarni
ng: invalid value encountered in true_divide
  return signal_softmax / (signal_softmax + background_softmax)
/home/surajrai1900/WatChMaL/analysis/classification.py:497: RuntimeWarni
ng: invalid value encountered in true_divide
  return signal_softmax / (signal_softmax + background_softmax)
```







for fixed 20% gamma mis-ID

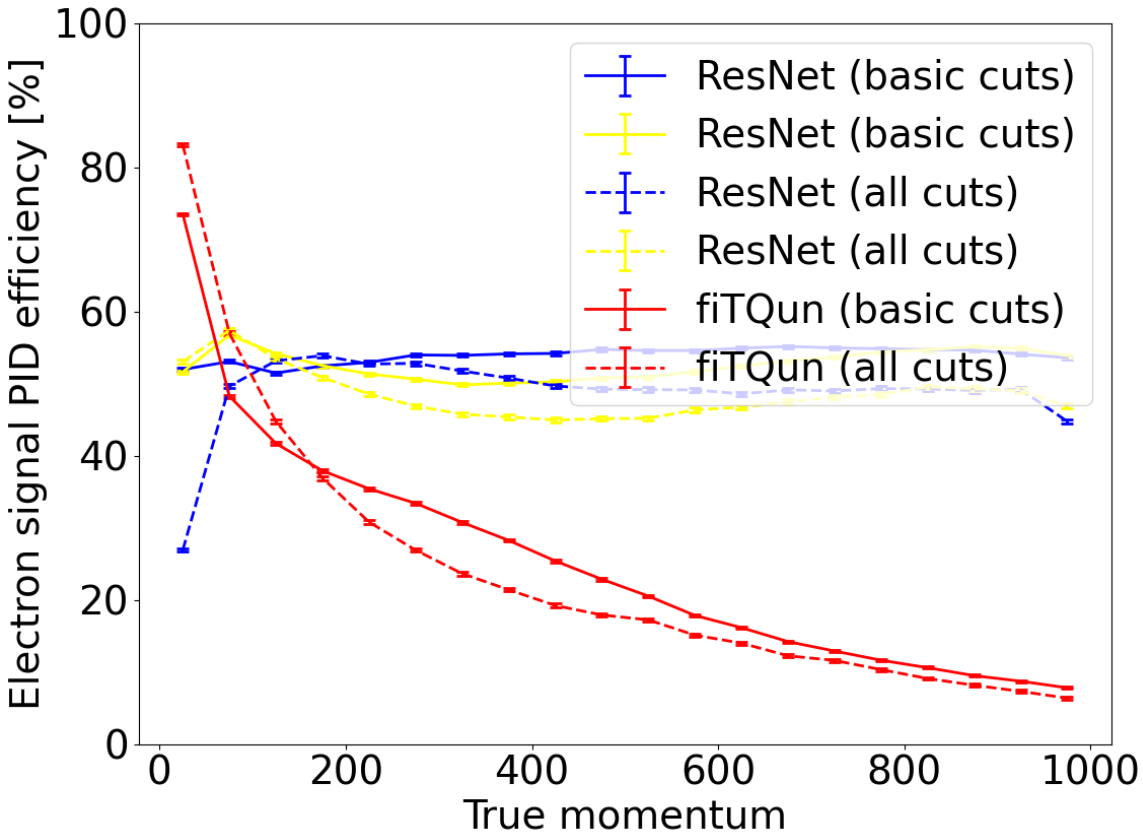
for each 50 MeV bin of reconstructed momentum, calculate the thresholds that reject 99.9% of muons and apply cut to all events

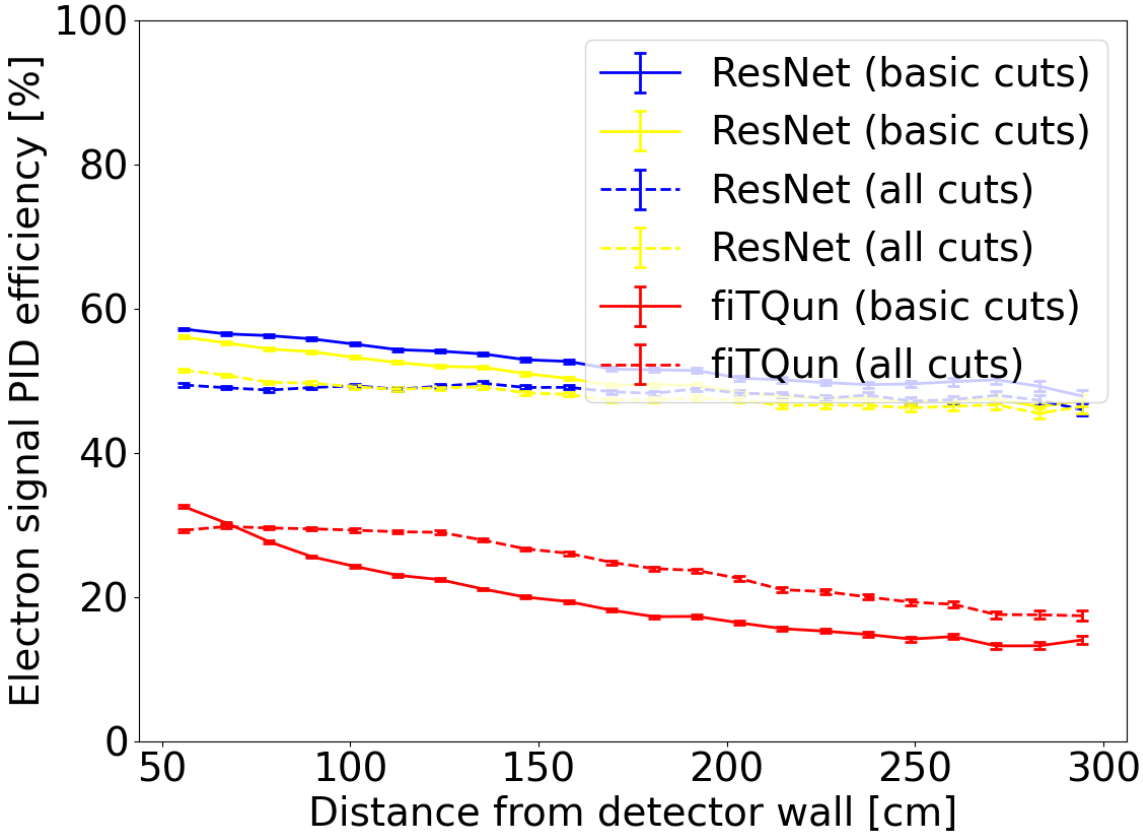
In [44]:
```
gamma_rejection = 0.8
gamma_efficiency = 1-gamma_rejection
for r in pid_runs:
    r.cut_with_fixed_efficiency(e_label, g_label, gamma_efficiency, selec
```

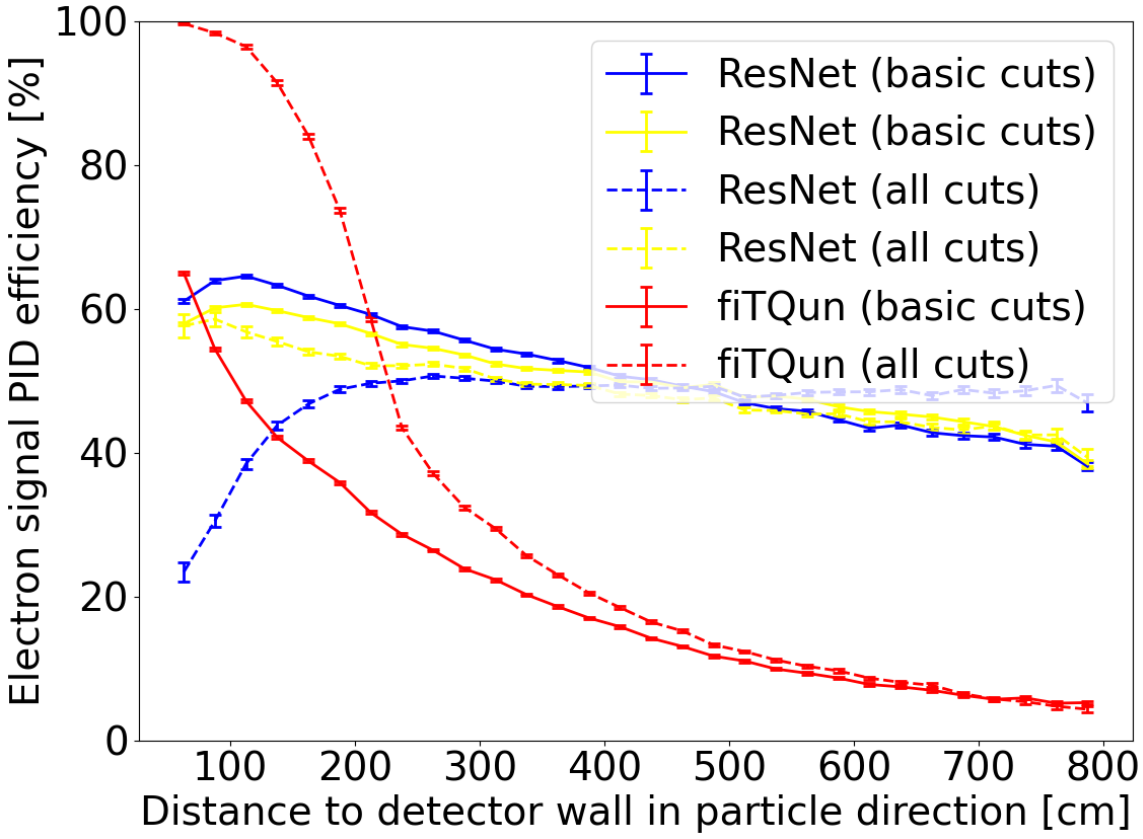
```
/home/surajrai1900/WatChMaL/analysis/classification.py:497: RuntimeWarni
ng: invalid value encountered in true_divide
  return signal_softmax / (signal_softmax + background_softmax)
```

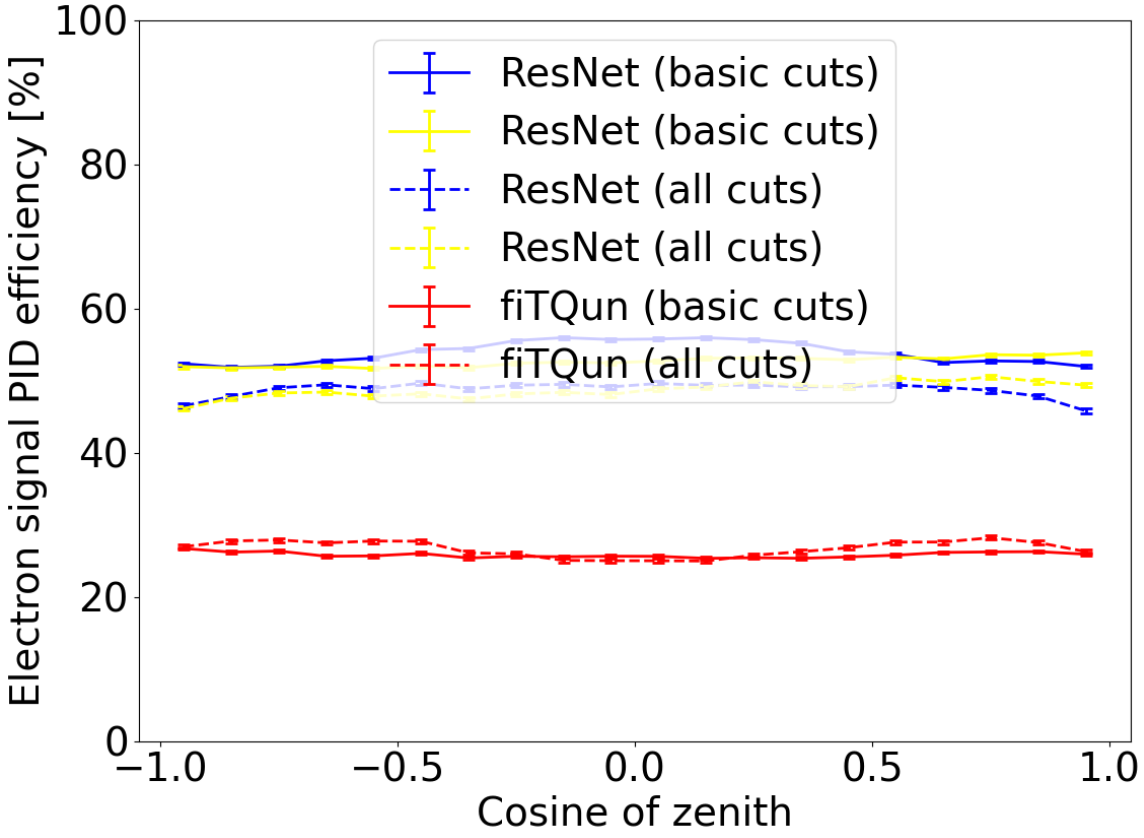
Plot signal efficiency against true momentum, dwall, towall, zenith, azimuth

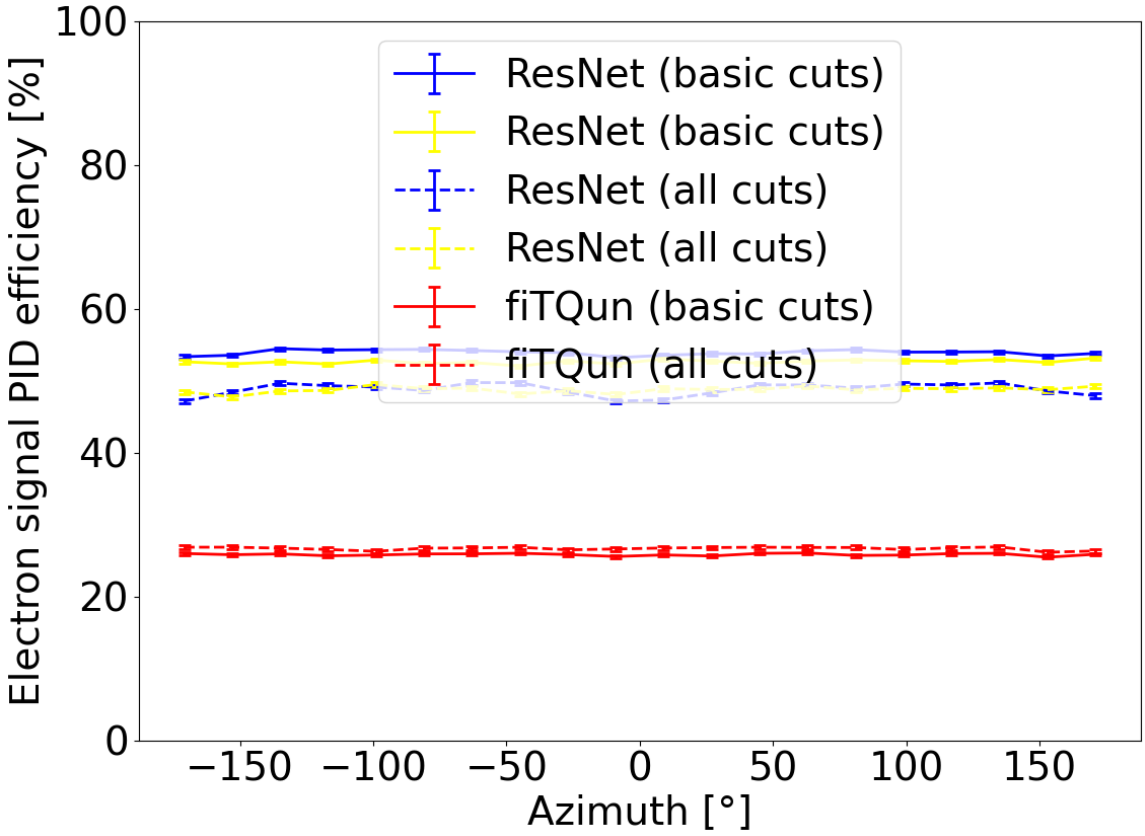
In [45]:
```
fig, ax = plot_efficiency_profile(pid_runs, mom_binning, select_labels=e_
fig, ax = plot_efficiency_profile(pid_runs, dwall_binning, select_labels=
fig, ax = plot_efficiency_profile(pid_runs, towall_binning, select_labels
fig, ax = plot_efficiency_profile(pid_runs, cos_zenith_binning, select_la
fig, ax = plot_efficiency_profile(pid_runs, azimuth_binning, select_label
```




Electron signal PID efficiency [%]
Distance from detector wall [cm]
ResNet (basic cuts)
ResNet (basic cuts)
ResNet (all cuts)
ResNet (all cuts)
fiTQun (basic cuts)
fiTQun (all cuts)


Electron signal PID efficiency [%]
Distance to detector wall in particle direction [cm]
ResNet (basic cuts)
ResNet (basic cuts)
ResNet (all cuts)
ResNet (all cuts)
fiTQun (basic cuts)
fiTQun (all cuts)

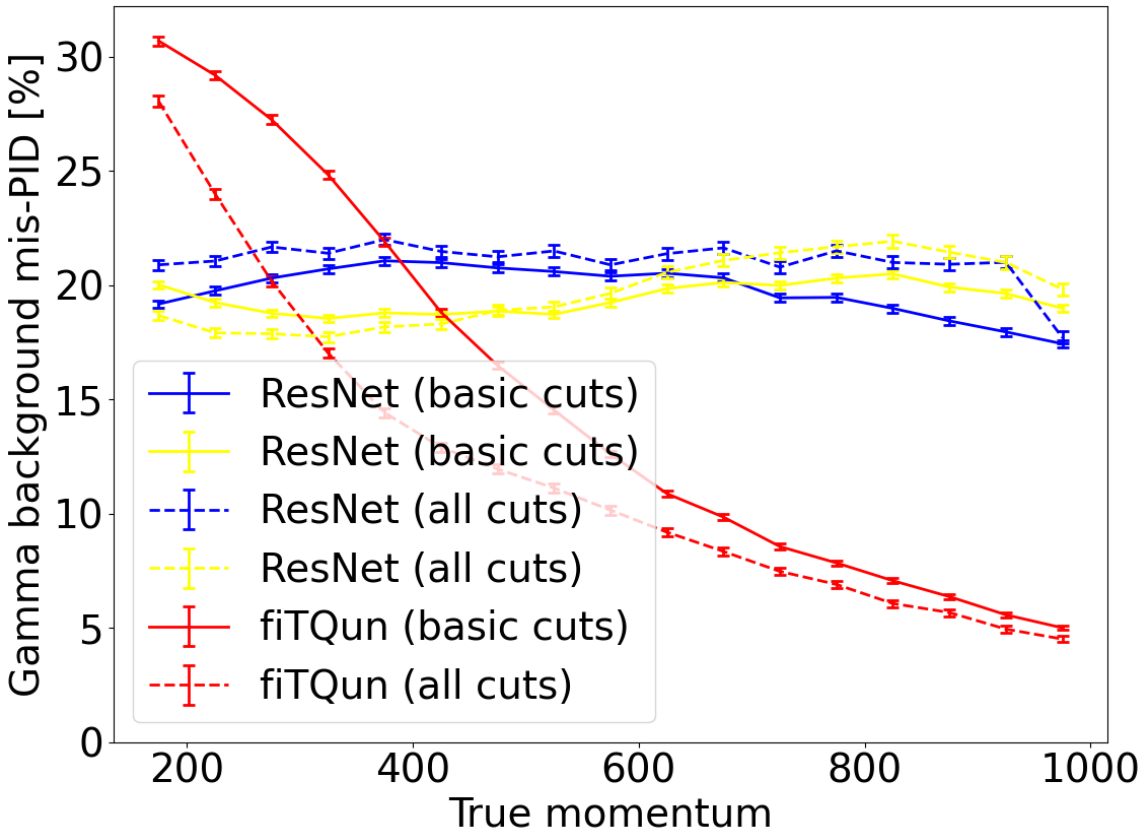


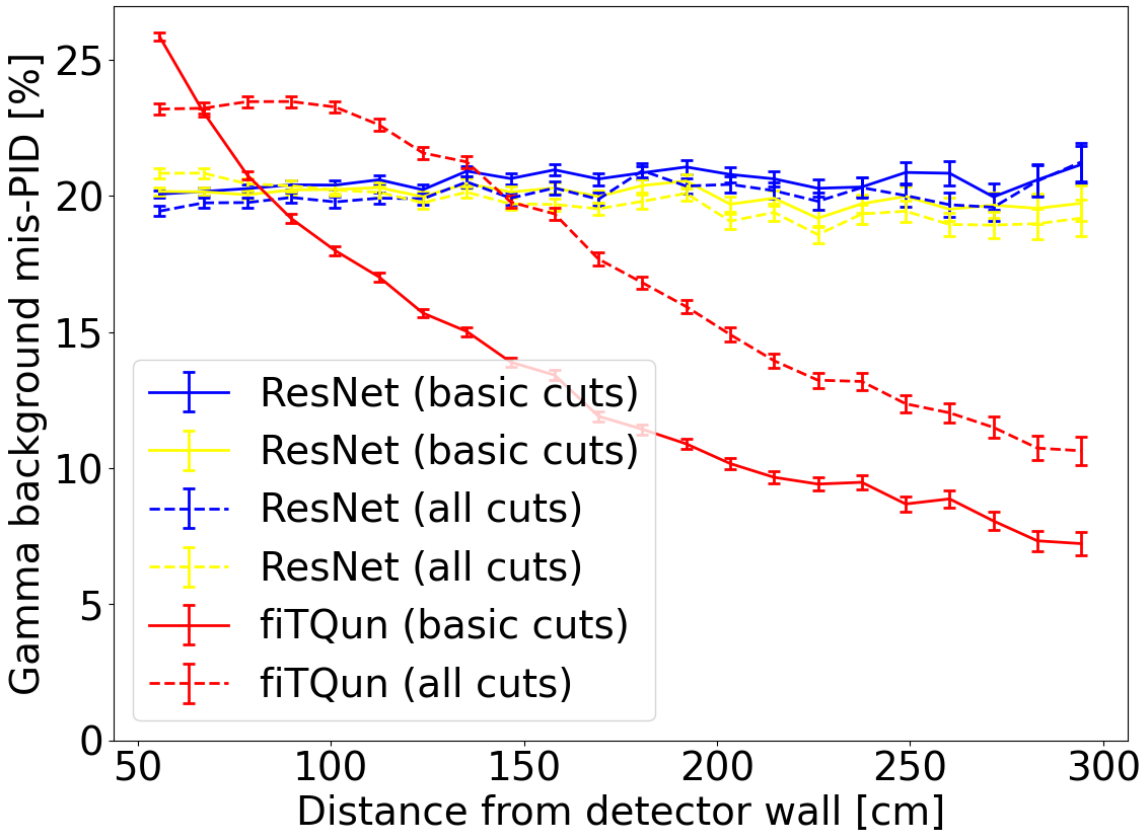


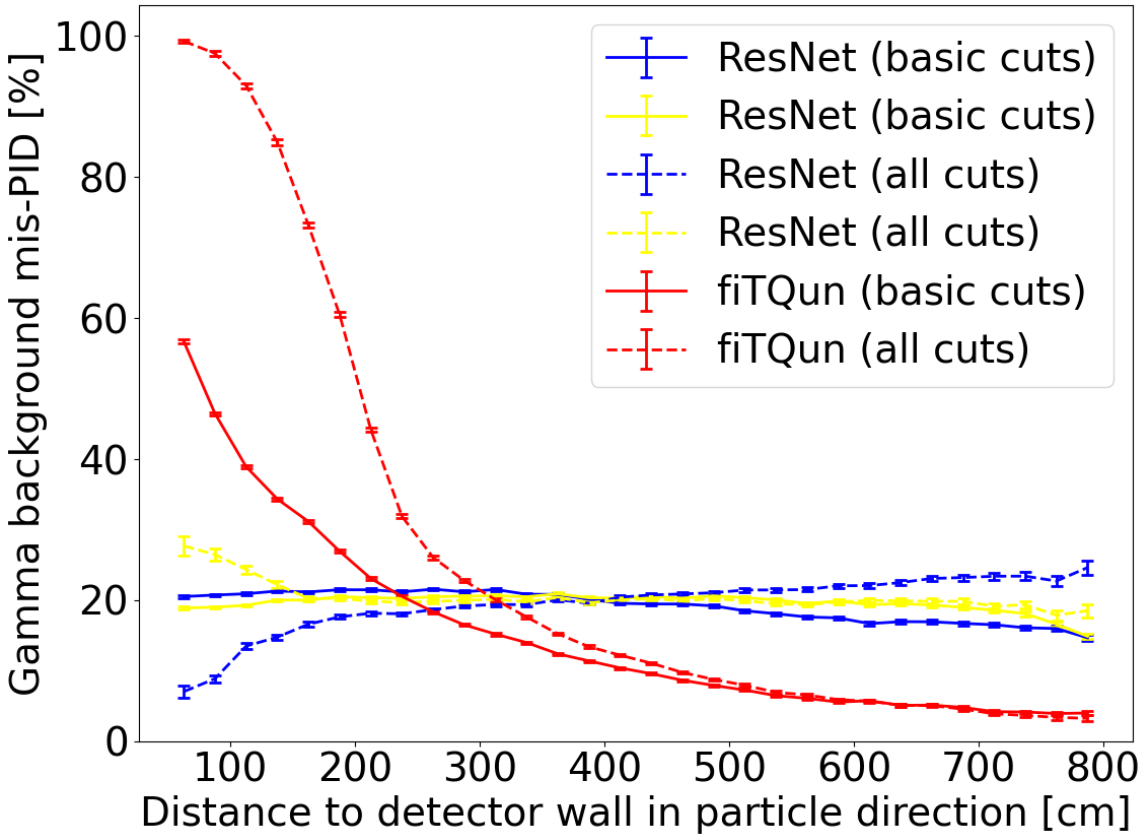
Plot background mis-ID against true momentum, dwall, towall, zenith, azimuth

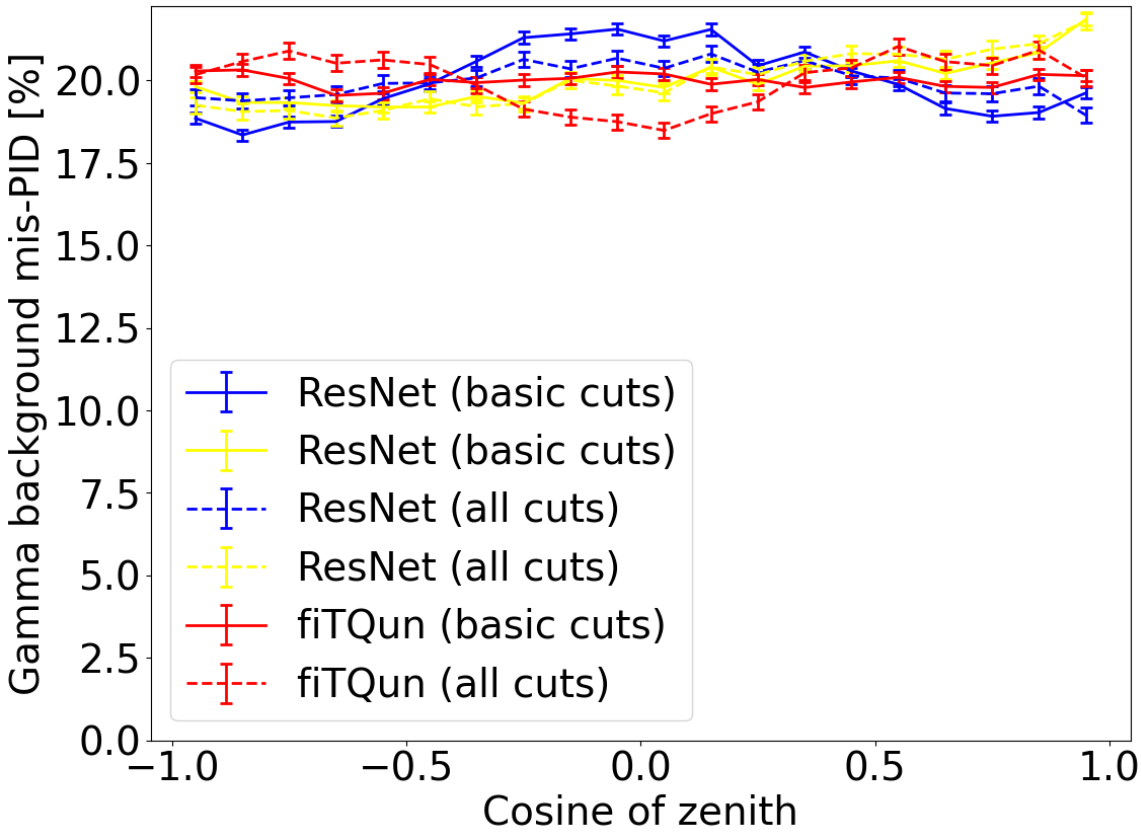
In [46]:
```
fig, ax = plot_efficiency_profile(pid_runs, mom_binning_mu, select_labels
fig, ax = plot_efficiency_profile(pid_runs, dwall_binning, select_labels=
fig, ax = plot_efficiency_profile(pid_runs, towall_binning, select_labels
```

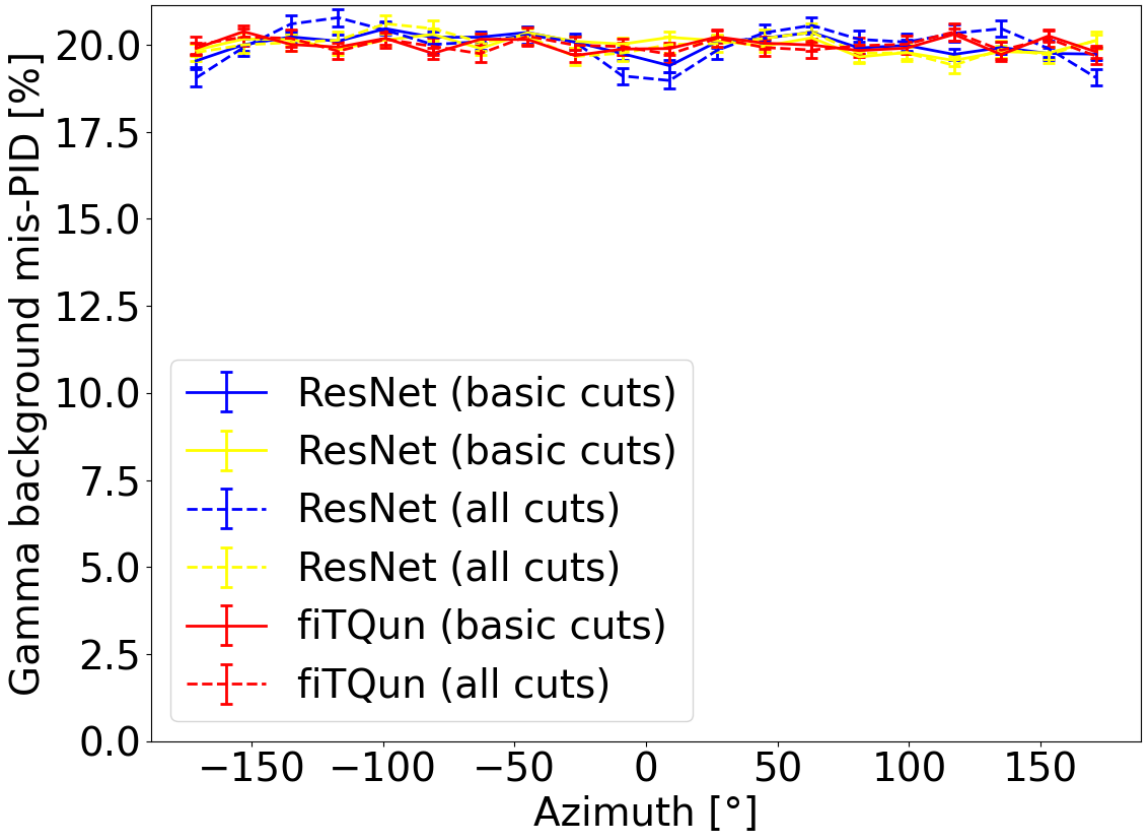
```
fig, ax = plot_efficiency_profile(pid_runs, cos_zenith_binning, select_la
fig, ax = plot_efficiency_profile(pid_runs, azimuth_binning, select_label
```






Gamma background mis-PID [%]
Distance to detector wall in particle direction [cm]
ResNet (basic cuts)
ResNet (basic cuts)
ResNet (all cuts)
ResNet (all cuts)
fiTQun (basic cuts)
fiTQun (all cuts)


Gamma background mis-PID [%]
Cosine of zenith
ResNet (basic cuts)
ResNet (basic cuts)
ResNet (all cuts)
ResNet (all cuts)
fiTQun (basic cuts)
fiTQun (all cuts)


Gamma background mis-PID [%]
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
−150
−100
−50
0
50
100
150
Azimuth [°]
ResNet (basic cuts)
ResNet (basic cuts)
ResNet (all cuts)
ResNet (all cuts)
fiTQun (basic cuts)
fiTQun (all cuts)