1. Eingabevariablen werden überprüft: also diese zb überhaupt gesetzt sind (deep\_models etc.)
2. Imagevariable wird erzeugt: checke die Länge des deep\_models (37-47)
3. Checke irgendwelche ortogh\_options (die offenbar gesetzt werden können) (48-55)
4. Checke auf Multinomiale oder Multinoulli ist (56-58)
5. Checke ob Daten liste einer Liste sind (59-62)
6. Checke ob formula richtig eingegeben ist (63-65)
7. Checke ob additional\_processors angegeben sind (66-68)
8. Setze die weight\_options: training mse
   1. Checke ob die Anzahl der weight\_ooptions die gleiche ist wie die Anzahl der list\_of\_formulas und setze is.lfun auf eine loss funktion (69-71)
   2. Wenn das nicht der fall ist (bzw wenn is.lfun nicht existiert) dann setze weight\_options anders (72-80)
9. Setze so <- penalty\_optiones (Zeile 83)
10. Setze penalty\_options (84-111): parse formulas
    1. $gamdata mittels Precalc\_gam: precalculate all gam parts from the list of formulas (in github unter psplinelayer)

Input: list\_of\_formulas, data, penalty\_options

Output: link every unique gam term to formula entries and the respective data transformation function

* 1. $with\_layer
  2. $weight\_options

1. Setze parsed\_formulas\_contents: enthält infos zu den Eingabeformeln, also infos zu den penalty\_options, output\_dim, additional processors usw. (wird in Schritt 10 berechnet), wird wie list\_of\_formulas benannt (111)
2. Checke ob subnetwork\_builder keine Liste ist (116-124)
3. Checke ob so$gamdata vorhanden ist (127-129): create gam data input
   1. Setzte gaminputs mittels der Funktion makeInputs() -> keras (subnetwork\_init)
4. Setze die additiven prädiktoren pro Formeleintrag von list\_of\_formulas: speichere diese in additive\_predictors (130-137)
5. Falls so$gamdata nicht vorhanden ist: setze die additive\_predictors anders (mit NULL) (138-145)
6. Bilde das Model mittels model\_builder() und speichere es in model ab (148-149) -> keras: model\_builder : which function to use for model building -> keras\_dr
7. Erstelle die variabel ret: darin wird alles gespeichert was von der Funktion deepregression() zurück gegeben werden soll (152-157)
   1. list\_of\_formulas
   2. gamdata
   3. additive\_predictors
   4. parsed\_formulas\_contents
   5. y
   6. ellipsis
   7. family
   8. penalty\_options
   9. orthog\_options
   10. image\_var
   11. prepare\_y\_valdata
8. FERTIG

Fragen:

1. Wie funktioniert die model\_builder() -Funktion? Mittels keras…
2. Wie funktioniert die makeInputs() -Funktion? Mittels keras…
3. Precalc\_gam müssen wir nicht verändern, oder?
4. Subnetwork\_builder[[i]] gibt Tensoren für NN aus?
   1. Outputs über tf$keras$layers, hier dann torch nutzen
      1. summary\_layer fügt die dann zusammen?
   2. orthog\_options = orthog\_control(orthogonalize = F) dürfen wir annehmen?
5. Habt ihr irgendwelche Tests für die Funktionen innerhalb von deepregression() geschrieben um unsere Funktionen vergleichen zu können?
6. Wie weiter machen? Ist es clever die Funktionen herauszusuchen die keras verwenden und diese dann ins Große Ganze einzuordnen und zu übersetzen in Torch?
7. Ist es schlimm wenn wir beim durchlesen nicht exakt alles verstanden habe? (Ich denke den groben Ablauf habe ich verstanden)
8. Nachdem wir die Funktionen die Keras verwenden ausgemacht haben, sollten wir dann die Funktionen aufteilen wer was übersetzt?